COUNTERING NETWORK LEVEL DENIAL OF INFORMATION ATTACKS USING INFORMATION VISUALIZATION

A Dissertation
Presented to
The Academic Faculty

by

Gregory John Conti

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Computing

Georgia Institute of Technology
May 2006

COUNTERING NETWORK LEVEL DENIAL OF INFORMATION ATTACKS USING INFORMATION VISUALIZATION

Approved by:

Dr. Mustaque Ahamad, Advisor
College of Computing
Georgia Institute of Technology

Dr. Henry Owen
Department of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Wenke Lee
College of Computing
Georgia Institute of Technology

Dr. John Stasko
College of Computing
Georgia Institute of Technology

Dr. Ralph Merkle
College of Computing
Georgia Institute of Technology

Date Approved: 23 March 06

To my mother and father, for supporting the purchase of my first computer.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support and guidance of many people. First and foremost, I would like to offer my sincere thanks to Mustaque Ahamad for serving as my advisor. He gave me a wide lane to explore, but his gentle hand kept me continually moving forward. I cannot fully express my appreciation to him for his mentorship and friendship. I also owe a large debt of gratitude to the other members of my committee: Wenke Lee, Ralph Merkle, Henry Owen and John Stasko. Each, despite the rigors of very busy schedules, gave freely of their time and expertise. Wenke brought me to a new level of network security understanding and taught me to leverage machine processing in ways that complement human analysis. Ralph helped me think about the long-term impact of my work and embodied the excellence that can be achieved through a lifelong career of scientific research. Henry provided continual support and guidance often taking on the role of devil's advocate. Many years ago, he first opened my mind to the opportunities available at Georgia Tech. John's enthusiasm for teaching and depth of experience in information visualization ignited in me a passion for the field of security visualization. His excellent course in information visualization was a seminal moment in my doctoral program.

Beyond my committee, many individuals and organizations provided valuable feedback and support of my work. I would like to thank: 2600, Kulsoom Abdullah, Sandip Agarwala, Jean Blair, Tom Cross, David Dagon, DEFCON, the Georgia Tech Denial of Information Research Group, Ron Dodge, EliO, Jeff Gribschaw, Julian Grizzard, the Georgia Tech Information Security Center, Hacker Japan, Mike Hamelin,

the Honeynet Project, the IEEE Information Assurance Workshop, Interz0ne, Kenshoto, Oleg Kolesnikov, Sven Krasser, Chris Lee, John Levine, Ling Liu, Michael Lynn, David Maynor, Jeff Moss, NETI@home, Ed Omiecinski, Calton Pu, Dan Ragsdale, Clark Ray, Gene Ressler, Rockit, Andre Sayles, Charles Robert Simpson, the Symposium on Usable Privacy and Security, Jason Spence, Carol Spisso, StricK, the United States Military Academy Department of Electrical Engineering and Computer Science, the United States Military Academy Information Technology and Operations Center, the Workshop on Visualization for Computer Security, Grant Wagner and the Yak.

I would like to highlight the contributions of my two Ph.D. program coordinators: H. Venkateswaran and Leo Mark. Venkat was my initial point of contact within the College of Computing and his warm welcome was a major contributing factor to my decision to attend Georgia Tech. Leo Mark continued this positive first impression by supporting every aspect of my program. The staff of the College of Computing contributed to this dissertation in many ways behind the scenes. I would like to thank Barbara Binder, Deborah Mitchell, Mike Nelson-Palmer, Mary Claire Thompson, Linda Williams and Becky Wilson for their support.

Finally, I would like to thank my family. My parents were instrumental in helping prepare me to undertake this work. They helped nurture my true passion for computing and unconditionally supported my endeavors, sometimes at the expense of their own. I would also like to thank my brother for setting such an adventurous and creative example through his own life. It pushed my thinking beyond equations and lines of code toward blending art with science. Most of all I would like to thank my wife and daughter. They both gave me unending support and lovingly tolerated my long hours of work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DOI	Denial of Information
DOS	Denial of Service
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
SSH	Secure Shell
SMTP	Simple Mail Transport Protocol
SNMP	Simple Network Management Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VoIP	Voice over Internet Protocol

SUMMARY

We are besieged with information every day, our inboxes overflow with spam and our search queries return a great deal of irrelevant information. In most cases there is no malicious intent, just simply too much information. However, if we consider active malicious entities, the picture darkens. Denial of information (DoI) attacks assail the human *through* their computer system and manifest themselves as attacks that target the human's perceptual, cognitive and motor capabilities. By exploiting these capabilities, attackers reduce our ability to acquire and act upon desired information. Even if a traditional denial of service attack against a machine is not possible, the human utilizing the machine may still succumb to DoI attack. When successful, DoI attacks actively alter our decision making, often without our knowledge.

In this dissertation, we address the problem of countering DoI attacks. We begin by presenting a taxonomy and framework of DoI attacks and countermeasures to add structure to the problem space. We then closely examine the use of information visualization as a countermeasure. Information visualization is a powerful technique that taps into the high bandwidth visual recognition capability of the human and is well suited to resist DoI attack. Unfortunately, most information visualization systems are designed without a clear emphasis on protecting the human from malicious activity. To address this issue we present a general framework for information visualization system security analysis. We then delve deeply into countering DoI in the network security domain using carefully crafted information visualization techniques to build a DoI attack resistant security visualization system. By creating such a system, we raise the bar on adversaries

who now must cope with visualization enhanced humans in addition to traditional automated intrusion detection systems and text-based analysis tools. We conclude with a human-centric evaluation to demonstrate our system's effectiveness.

CHAPTER 1

INTRODUCTION

The Problem of Denial of Information Attacks

We experience denial of information (DOI) attacks every day. Our inboxes are swamped with spam and the subject lines sometimes fool us. Our search engine queries return many irrelevant results and online auctions are plagued by corrupt database records filled with intentionally misleading keywords. We waste time sifting through useless information and in many cases we are actively tricked into taking incorrect actions. These attacks go far beyond simple denial of service (DOS), which attempt to prevent legitimate users access to a service and typically target the processing, memory and bandwidth resources of the machines we use. While denial of information attacks include denial of service, they extend to include any attacks that intentionally or unintentionally consume human resources, mislead, confuse or trick the user into taking inappropriate actions or not taking appropriate actions. Ultimately, denial of information attacks target the human. The result may simply be consumption of time and the slowing down of our decision-making or, even worse, force us to make incorrect decisions.

Concealing of information (steganography) or attempting to make it unreadable without special knowledge (encryption) are legitimate related issues, but are distinct from denial of information. The key difference is that denial of information attacks attempt to manipulate or overwhelm the human with malicious information thereby limiting their ability to get to the information they seek.

The problem of denial of information attacks is not new. History provides a rich set of examples that illustrate the long running quest to find the right information at the right time

despite a great deal of noise within the information environment. Examples include the failure of the United States to predict the attack on Pearl Harbor on December 7, 1941 and on the World Trade Center on September 11, 2001.

This quest is exacerbated today by the rapid growth in the rate at which information is generated and communicated. Two recent University of California, Berkeley studies on how much information is created clearly illustrate the problem [71,72]:

- In 2002, about 5 exabytes of new information were created in print, film, magnetic and optical formats. In 2000, the estimate was only one to two exabytes per year. Five exabytes is equivalent to 37,000 times the size of the United States Library of Congress book collection or 800 megabytes per person based on the world population.
- From 1999 to 2002 information in these formats grew at a rate of 30% per year.
- Ninety-two percent of this information was stored on magnetic media.
- In 2000, humans annually exchanged some 610 billion emails and browsed 2.1 billion static web pages on the World Wide Web.
- Use of the Internet has grown 2000% from 1992 to 2000.

In today's information environment, this growth has outstripped our ability to find and process the information that we seek, despite the increased information processing capabilities at our command. This is a non-trivial problem even without being hampered by malicious entities seeking to inject noise into our environment. In this dissertation we assume that in a world without malicious entities traditional information retrieval and information overload issues can be worked out. These malicious entities are driven by a myriad of motivations, including

political, economic and social gain. In this dynamic environment it is difficult to even determine who the attackers are. One individual's absolutely credible source is another's propaganda, but we assume there is only one ground truth. The combination of intentional and unintentional clouding of our information environment prevents us from finding, assimilating and acting upon the information we seek. Representative examples include:

- The Google SafeSearch feature, while intending to filter sexually explicit content, actually filters a great deal more, including tens of thousands of pages without any such content [30].
- In the late 1990's a World Wide Web bookseller purchased the word "the" on a pay-per-click search engine, resulting in thousands of irrelevant responses to search queries.
- Web server content when filtered at the IP address level can block access to many other unintentional sites sharing the same hosting service [31].
- A Silicon Valley programmer became so enraged by the content of spam he received that he sent threats to torture and kill employees of the offending company. In an interview after his arrest, he "acknowledged that he had behaved badly, but said his computer had been *rendered almost unusable* for about two months by a barrage of pop-up advertising and e-mail." [125]

In comparison to traditional denial of service (DOS) attacks, denial of information (DOI) attacks target the human by exceeding their perceptual, cognitive and motor capabilities. Even if traditional DOS attacks against a machine can be countered, the human utilizing the machine to process information may still succumb to a DOI attack. The ultimate result of a successful attack is a human that does or does not take actions they otherwise would have. As a first step to

counter these attacks, we must develop mechanisms that will allow us to find the right kind of place in the information space, one that will provide the information that we seek. This can be visualized in figure 1.1.

In this figure, the large circle represents the total information sphere; information produced by all sources and all possible communication channels. This information universe is constantly evolving; new information is constantly being created while other information is continually being destroyed. Whether information is pushed to the human (as in an email list) or pulled (as in a web search engine) the human must attempt to select and properly address the information they seek (represented by the star). The smaller circle represents this selection and the lines linking the human to this circle represent the communication channel. Due to limitations in today's addressing mechanisms and semantic search technology, frequently this search will return unwanted information (noise) concealing the information they seek (signal).

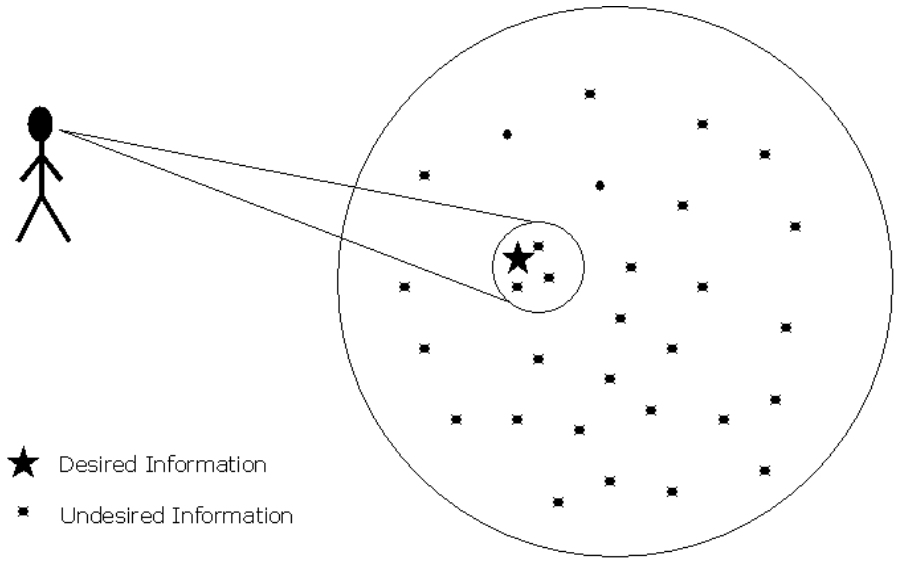


Figure 1.1: The Information Universe

Consider the signal-to-noise ratio of your email inbox, messages that you value are the signal and others are noise, spam. Other problems you might encounter include the lack of an answer in the information space or an inability to adequately specify the search such that it includes the desired information. To analyze the effect of the signal to noise ratio on the human, it is useful to consider the OODA loop model developed by United States Air Force Colonel John Boyd [53]. In this model, humans repeatedly go through four steps (Observe, Orient, Decide and Act) as they assimilate information and act upon it. In a competitive environment the individual who can execute these steps faster has the advantage. Each step can be slowed by denial of information attacks. For example, an email user may spend time deciding to delete a given email with a particularly compelling subject line. Another class of attack will force the user to perform extra cycles through the loop. For example, an email user must spend time scanning their inbox, deciding to delete (or not delete) each message and then physically performing the delete operation. Ultimately, these attacks might exhaust the human's available resources (e.g. time for the task) so that the loop halts as the human moves on to other tasks. A third class of attack includes those that so cloud the information environment that incorrect decisions are made, also forcing extra cycles through the loop.

There are potentially many ways to help counter denial of information attacks. In Chapter 3 we will cover a broad range of these techniques. In later chapters, we focus on the use of information visualization to address denial of information problems associated with network security. The massive amount of security data generated by network sensors and host-based applications can quickly overwhelm the operators charged with defending the network. Often, important details are overlooked and it is difficult to gain a coherent picture of network health and security status by manually traversing textual logs, using command line analysis scripts or

traditional graphing and charting techniques. In many instances, this flood of data will actually reduce the overall level of security by consuming the operator's available time or misdirecting their efforts. In extreme circumstances, the operators will become desensitized and ignore security warnings altogether, effectively negating the value of their security systems.

Using Information Visualization to Counter Denial of Information Attacks in the Network Security Domain

In this dissertation we focus specifically on this denial of information problem by carefully crafting an information visualization system designed to present security data in insightful ways that tap into the high bandwidth visual recognition capability of human operators. To actively counter denial of information attacks against security analysts we constructed and evaluated an experimental system to improve data quality and analyst insight. The visualization techniques we used reduce the amount of irrelevant data (noise) and increase the useful information (signal) presented to the analyst.

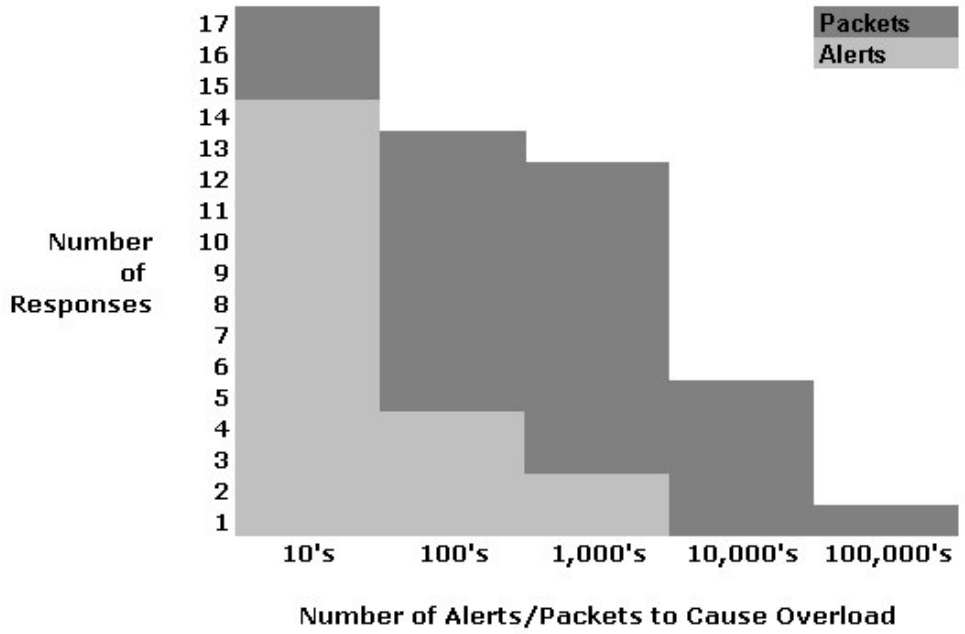


Figure 1.2: Results from Survey of Expert Security Analysts from Academia and Industry. Figure depicts threshold at which human overload occurs when analyzing network packets (dark gray) and intrusion detection system alerts (light gray).

We began our work by surveying security analysts to determine the limits of today's best systems and identify high payoff targets for improvement. There were 39 participants, primarily professional security analysts working in industry. Our survey studied the limits of two open source security tools in very wide use: Snort, an intrusion detection system, and Ethereal, a network analysis tool. Figure 2 shows that the majority of the survey participants start to become overwhelmed when the number of intrusion detection alerts reaches only tens of alerts per hour. Similarly, in packet analysis, overload occurred when faced with only hundreds to thousands of packets. Participants were explicitly asked to describe when they, as humans, became overloaded and not to respond based on the resource limitations of their given hardware platforms. It is useful to examine the thresholds by analyst background. Information security students had a lower capability to cope with alerts, reaching overload at an average of 30 alerts

per hour. Professional security analysts were able to handle more alerts than students, but most still faced overload at approximately 230 alerts per hour. The situation is similar for packet analysis. Information security students felt that Ethereal became difficult to use when the number of packets exceeded 500. Professional security analysts were able to handle a larger number of packets without overload, but most respondents felt that Ethereal became difficult to use when faced with datasets of more than 6,000 packets. Our security visualization system directly addresses these issues and our results indicate that analysts using it are able to cope with approximately ten times more information without overload and can rapidly detect and locate data of interest that would be extremely difficult, if not impossible, using conventional tools.

Dissertation Overview

Thesis

Our thesis is that it is possible to counter denial of information attacks in the network security domain by augmenting humans with carefully crafted visualization systems.

Thesis Contributions

In this thesis we make contributions in the following areas in direct support of our thesis.

Taxonomy and Framework of Denial of Information Attacks

The problem of denial of information attacks is ill defined. Chapter 3 addresses this problem from a big picture perspective and contains the following contributions:

- a taxonomy of denial of information attacks and countermeasures
- technology independent principles for designing systems that will resist attack

Information Visualization Threat Analysis

Before information visualization systems can be created to counter denial of information attacks, a clear understanding of possible threats and countermeasures is required. Chapter 4 performs such an analysis and makes the following contributions.

- a framework for information visualization system security analysis
- a taxonomy of malicious attacks
- technology independent principles for designing information visualization systems that will resist attack

Visualization System

Chapter 5 consists of the results and lessons learned from the design and implementation of a security visualization system based upon the counter DOI principles discussed in Chapters 3 and 4. The primary contributions fall into the areas of system design, attack fingerprinting and task specific visualization design.

Attack Fingerprinting

In order for security visualization systems to be effective, it is critical that the analyst can identify events of interest. While attacks have evolved that evade traditional automated intrusion detection and analysis techniques, a visualization enhanced human will inherently detect different signatures and different anomalies. While it is still possible to fool a network analyst or system administrator, we argue that properly designed visualizations enhance the capabilities of the human in such a way that greatly complicates the efforts of an attacker. The fingerprinting section makes the following contributions.

- demonstrates that commonly employed network attack tools leave clear visual fingerprints
- presents a catalog of network attack tool fingerprints

Task Specific Visualization Design

Tightly coupled animated security visualization**:** Current work in the area of security visualization primarily uses stand-alone visualization windows and loosely coupled static displays. Our visualization technique tightly couples three windows to provide a flexible, animated display of network traffic to allow insightful monitoring and playback of captured network traffic. The primary contributions described in this section include:

- the novel use of tightly-coupled, animated, time-sequence scatter plots and parallel coordinate plots in both 2D and 3D to rapidly analyze network traffic
- an exploration of the effective use of labeling, animation, scaling, and fading as well as interaction techniques to cope with extremely large ranges of categorical and discrete numeric data
- an in-depth analysis of real-time and forensic network traffic from a large scale university honeynet
- the ability to scan large datasets of network traffic for malicious activity despite the visual noise of legitimate activity and facilitates "at a glance" insight directly supporting network monitoring, intrusion detection and attacker behavior determination

Malicious Binary Object Visualization*:* While visualization of network security data has recently emerged as a powerful technique to provide insight and support analysis, current best practices have operated at a high-level and examined only external information, such as packet addressing information and miss payload based malicious activity. The primary contributions of this section cover the following:

- a novel visualization technique that allows comparison of hundreds of binary objects
- experimental results from using the technique on malicious and non-malicious network traffic datasets
- a novel application of the semantic zoom paradigm in the security visualization domain

System Design

Visualization techniques do not stand alone. They require an underlying system architecture and interface to be effectively used. We created such a system by integrating the novel visualization techniques discussed above with five other visualization techniques that have rarely been applied to the network security domain. We crafted task-specific variants of these visualization windows for a total of 15 different views and integrated them all into a single, easy to use system using a PVR metaphor. The key contributions of this section include:

- a framework for security visualization systems design
- lessons learned from the design and implementation of our security visualization system
- the novel use of the PVR metaphor as the core user interface

Human-centric Evaluation of Security Visualization Systems

To evaluate the effectiveness of our security visualization system we conducted two controlled experiments to test the ability of intermediate and advanced users to both detect and precisely locate security anomalies in network traffic. The primary contributions include:

- the first quantitative evaluation of a security visualization system
- results and lessons learned from the design and execution of the experiments
- a quantitative validation of our system and visualization design

Thesis Outline

The remainder of this dissertation is organized as follows. Chapter 2 places our work in the field of current research and synthesizes the state of the art in denial of information and the visualization of security data. Chapter 3 presents a taxonomy of denial of information attacks and countermeasures in order to add structure to the problem. Section 4 presents a general framework for information visualization system security threat analysis, a taxonomy of visualization attacks and technology independent principles for information visualization system designers. Chapter 5 includes the results and lessons learned from the design of our DOI resistant security visualization system and a general framework for the design of future systems. Chapter 6 describes and analyzes the results from two human-centric evaluations of our system and visualization techniques. Chapter 7 presents our conclusions and directions for future work.

CHAPTER 2

RELATED WORK

The area of denial of information is intensely multi-disciplinary; spanning, at least, computer science, communications theory, knowledge management, intelligence analysis, epistemology and even library science. In this dissertation, we examine denial of information initially from the big picture perspective and then deeply explore the problem in the network security domain. This related work chapter follows the same structure. We will first place our work amongst the larger problem of denial of information and then focus specifically on the use of security visualization as a solution. In the area of security visualization we examine the key aspects: attacking visualization systems, system design, visualization design, user needs assessment and human-centric evaluation in order to provide the context into which our work is placed.

Denial of Information

Denial of Information is a broad problem, but two areas, in particular, illustrate current best practices: spam and search engines. Techniques for countering spam and improving search engines deal directly with the core problem of denial of information by attempting to increase signal while reducing noise.

Spam illustrates denial of information attacks using push technology. Email is pushed from an information producing node to the inbox of the human. A human's inbox can quickly become flooded with useless and resource consuming unsolicited email. Spam is fought with technology, the legal system and by humans forming counter-spam organizations. The first-ever spam conference was held in 2003 [98]. Also in 2003, the United States Federal Trade Commission hosted a widely publicized forum to explore potential solutions to spam [115]. The

open source community has banded together to create SpamAssassin [97] which uses email header analysis, textual analysis, the Vipul's Razor spam signature database [117] and blacklists such as mail-abuse.org [73] and the Open Relay Database [81] to block spam. Beyond the more traditional keyword and text-based analysis filtering, Bayesian filtering has emerged as today's counter-spam technology of choice [48,49,86,92]. Burton's SpamProbe program currently reports 99.96% effectiveness using multi-word Bayesian filtering, but spammers are adapting their techniques in an attempt to overcome Bayesian filters [99]. Due to ongoing one-upmanship, it is uncertain whether Bayesian filtering will continue to enjoy this level of success in the long run. Legal attempts to counter-spam are also being made. The Federal Government of the United States recently passed legislation that prohibits sending commercial email messages unless the recipient has requested it [58]. It remains to be seen if legal action will be effective in reducing spam. More aggressive measures are also being suggested. Graham suggests using web-spiders to extend the reach of current filters by scanning links within email. Given the high volume of spam sent, this could result in a DOS attack against the spammer. Graham also suggests the potential of an ELIZA-like artificial intelligence program to engage spammers via email [50]. This is particularly interesting because it turns the tables against the attackers and performs a DOI counterattack against the human aggressor.

The corruption of search engines illustrates denial of information attacks using pull technology. The user formulates a query and pulls information from a search engine. If a web search engine's database has been corrupted, and all of them are to some degree, the user will be subject to a denial of information attack. Initially, search engines relied upon HTML meta-keyword and meta-description information as well as full text searches of pages to construct their indices. Web developers soon discovered attacks that degrade the effectiveness of these

techniques [102,103]. Currently, many search engines rely on the democratic nature of the web's hyperlink structure to build relevant database indices. Representative of this type of denial of information countermeasure is Google's PageRank system [45,84]. PageRank calculates authoritative sources of information and gives them better placement in the search engine. Authoritative sources are identified by a large number of incoming links from external pages. The PageRank algorithm increases the weight of these external pages based upon their importance. It is supplemented with Googlebot technology that verifies the freshness and accuracy of its search index [46]. Web indices such as Yahoo rely upon human web surfers to verify the veracity of initial listing requests and updates [126]. Despite the effectiveness of systems such as PageRank there is a continuing effort to subvert them [23].

Other pertinent research includes the study of information design and information quality. For example, the information design work by Tonfoni analyzes the allocation and utilization of information resources despite incomplete information, limited time and cost concerns. Of particular note is her discussion on what is "highly significant" to the consumer of the information and what is "disturbing noise." [109] Researchers studying the area of information quality have similar concerns. For example, Wand uses an ontologically based approach to define data and improve overall quality. [119]

There are a wide variety of techniques from other domains that can potentially be used to counter denial of information attacks. The Public Key Infrastructure (PKI) uses a Certificate Authority (CA) to enable confidentiality, integrity and authentication for communications [28]. PKI, while generally improving trust, suffers from several shortcomings including duplicate names in large communities, failure of some components of the infrastructure to check for revocation of credentials, and the potential for theft of credentials due to system security failure

or compromise of personnel. Previous work by Ahamad presents an overview of the denial of information problem, as well as a node-based model for gathering feedback from users, based upon positive or negative transactions, to create private and shared trust information [2].

Trusted computing seeks to allow a user to trust their system and, importantly, allows others to trust your system as well. Trusted computing typically relies upon a chain of trust beginning with trust of hardware and commercial operating systems software, but there are many open questions and a final trusted computing solution lies beyond reach [4,118].

While there has been some work in the area of DoI there has not been a concerted effort to coalesce and define the field. Chapter 3 helps to address this issue, as well as extend our related work discussion, through the development of a comprehensive taxonomy of denial of information attacks and countermeasures with specific emphasis on technical solutions.

Attacking Visualization Systems

Visualization systems are employed to counter denial of information problems in many domains, but while each information visualization system and technique has inherent strengths and weaknesses (see [123] for an excellent survey) researchers do not examine the potential of a malicious entity acting upon the system. As we explore the construction of a security visualization system it is critical to perform a threat analysis in order to understand where it is vulnerable to attack.

The field of information warfare and the related fields of psychological warfare, propaganda and battlefield deception do include the notion of external malicious entities. In general, these fields seek to use deliberately false or misleading information to change people's understanding through deception and confusion rather than persuasion and understanding [29]. While the techniques of distraction, misinformation and disinformation are quite relevant, they have not

been widely used in the context of information visualization. We will consider these techniques in our work.

Information visualization, as an area, involves analysis of data sets in which the data is more abstract in nature, having no natural physical or geometrical representation [100]. Examples of data domains common to information visualization include statistics, financial data, text, and software. Research into the manipulation of information visualization systems is relatively uncommon, however. The VizLies special session of several IEEE Visualization conferences [60] did address malicious visualization, but only in an informal manner, as entertainment at evening social functions. Several researchers have more formally considered the notion of malicious visualizations. Tufte addressed such concepts as the "lie factor," disappearing baselines, the difference between height and volume in pictograms, misleading or missing scales, missing data, design dominating the data and the effect of 3D graphics on correct interpretation [111,113,114]. All are valid, but anecdotal, instances of malicious visualizations. Tufte further explores the boredom, wasted-time and degraded quality and credibility of communication by incorrectly utilizing PowerPoint presentation software [110,112]. While there are some interesting characteristics relevant to malicious visualizations (e.g. degraded quality of information and wasted time), these essays deal with the limitations of PowerPoint presentations in a non-interactive speaker to audience scenario. Books such as *How to Lie with Charts* [63] and *How to Lie with Statistics* [59] also explore techniques to design presentations and reports that mislead audiences or readers. In a similar vein, researchers such as Globus [41] and Bailey [7] focus on how system creators can massage their results to mislead audiences. Rogowitz considered the application of perceptual rules to preventing "lying with visualization." He did not consider external malicious entities [89].

From our perspective, the primary limitation of these works is that they focus on techniques the *creator* of the visualization system, business presentation, advertisement or statistical report can use to manipulate their audience. Our work assumes that this is not the case and that the creator of the information visualization system is non-malicious. *Our malicious entities attempt to attack the system itself, it's data and the human attempting to utilize it. They are not the owners or creators of the system in question.*

The uniqueness of our work, presented in Chapter 4, stems from the comprehensive analysis of the weaknesses of visualization systems, and their supporting data flow, including: data sources, data communications, data storage, processing, presentation and human interpretation. A novel taxonomy of attacks is presented as well as a technology independent set of design principles to assist in countering such attacks. We used these design principles to inform the design of our system presented in Chapter 5.

Security Visualization

We use visualization to counter denial of information attacks in the network security domain. To do this, we must be able to develop systems that allow us to detect malicious activity despite the noise of DoI. The following sections describe how our work extends current research in fingerprinting such malicious activity and packet level security visualization.

Fingerprinting

Related work falls into several main areas including current network security visualization research as well as application and operating systems fingerprinting. While network visualization and intrusion detection are relatively mature areas, there is a limited body of work covering network security visualization. Representative recent research includes analysis of the

stability of Internet routing[104,106], analysis of stepping stone pairs[105], monitoring the security status of large networks [93], mapping of the Internet[17,24], application of statistical methods for intrusion detection[74], intruder behavior characterization[33], worm propagation[22], rapid prototyping[64], TCP/IP sequence number generation[130,131], haptic integration [80] and the construction of a toolkit for visual intrusion detection [43]. Availability of security-centric commercial and open source/free visualization systems is likewise limited. Representative examples include: Secure Decisions' SecureScope, Lancope's StealthWatch + Therminator, Etherape, Netstumbler, 3DTraceroute and XTraceroute. SecureScope provides high-level overviews of intrusion detection alerts projected onto a map-like matrix. StealthWatch + Therminator applies thermodynamic theory to create a bar graph indicative of the quality of data at any given time and relies upon the StealthWatch intrusion detection system for source data. Etherape creates a circular graph of network nodes and creates interconnections based on network activity. Netstumbler uses a colored bar graph to show signal strength of wireless network signals. 3DTraceroute shows delays between network nodes in a three-dimensional plot and XTraceroute uses IP geolocation technology to plot traceroute data on a globe.

We chose a comprehensive approach to visualize network attacks that included consideration of all TCP, UDP, IP and Ethernet header fields as well as many features that can be derived from this data. After examining the data available, we considered a broad range of visualization techniques from classic information visualization literature and current research in network security visualization. Finally, we examined a variety of network attack tools from the Top 75 Network Security Tool List [38] produced by Fyodor, the creator of nmap. This list was constructed based on a May 2003 survey of nmap developers. From this consideration of data,

visualization techniques and security tools we constructed a series of experiments to test the hypothesis that these tools could be effectively fingerprinted. We understand that a proficient attacker can evade many of these techniques, primarily due to lack of authentication in today's network protocols, but feel that visual fingerprinting remains one of the most effective techniques at present. Such is the case with much of the information security field, for now and into the foreseeable future it will likely continue to be an on going battle of one-upmanship.

The primary contributions of this portion of the dissertation include the demonstration of the efficacy of fingerprinting common attack tools, the ability to provide rapid insight into the attacker's operating system type and the possible lineage of the code in use, the ability to detect some classes of stealthy attacks and the ability to detect slow scans despite the visual noise of legitimate traffic.

Combined Visualization

Information visualization is a mature field with a wide range of techniques that have been successfully applied to many domains. But only recently has work been done in earnest to apply these techniques to network security and other related information assurance problems. Examples include Girardin [40] who used self-organizing maps to detect malicious network activity and Nyarko [80] who used haptic feedback to find suspicious activity in network traffic. Another interesting example is the Spinning Cube of Potential Doom [69], which visualizes real-time port and IP data in a three-dimensional cube as a rotating scatter plot. While quite useful to see coarse trends in large-scale networks, it lacks animation, multiple visualizations and interactive capability. The visualization technique we outline in this portion of the dissertation combines animated scatter plots with parallel coordinate plots. The notion of parallel coordinate

plots was first proposed by Inselberg [61]. Several researchers have applied the technique in the network security domain, including Marchette [74] and the National Center for Advanced Secure Systems with their VisFlowConnect tool [93,127]. We extend their work by combining parallel coordinate views and animated scatter plots into a single cohesive visualization. In addition, we add three-dimensional functionality that allows the user to zoom and pan the combined visualization. We also explore system performance characteristics not seen in other work as well as the use of fading. Fading is used in Etherape, but the tool provides only a single visualization of network traffic arranged in a circular graph [34]. Similarly, Erbacher [33] used a circular visualization composed of animated glyphs to demonstrate that intruder behavior is surprisingly observable. Both of these approaches, while effective for moderate numbers of network nodes, can quickly become crowded. Our combined visualization approach supports a far greater number of nodes. We estimate this gain to be about an order of magnitude more nodes. When combined with our zooming capability, this gain can reach several orders of magnitude, but at the cost of additional human interaction. During the design of this portion of our system, we relied upon the work by Komlodi [66] and Fink [36] as well as our own survey (see Chapter 6) to ensure we incorporated real-world operator requirements. We also examined best of breed open source security visualization tools. Snort [96] and tcpdump [108] provide industry standard network monitoring capability, but only as textual output. Ethereal also offers excellent textual output and protocol parsing, but only provides very basic visualization capability [35]. In our research, we analyzed several interesting datasets including botnet and honeynet traffic. To the best of our knowledge, there has been no work done on visualizing botnets. There has been some initial work on honeynet visualization. Honeynet monitoring is typically conducted with ethereal, snort and tcpdump, but there are two tools, primarily text-based, that provide some

visualization capability. The Honeynet Security Console is a useful tool to store and analyze large amounts of data [1]. Sebek can correlate low-level host-based data (e.g. launched programs, keystrokes, accessed files) with network data to evaluate honeynet intrusions [8]. The result can be visualized as a dependency graph. Another useful, but again, primarily text based tool is OSSIM. Its primary strength lies is the correlation of multiple data streams, a feature that we do not currently provide [82].

Binary object visualization

Visualization of security data has recently emerged as a powerful technique to provide insight and support analysis that is difficult with traditional text and charting techniques as well as signature and anomaly based machine processing. Current best practices in the network visualization domain employ scatterplots [42,75,127], parallel coordinate plots and line segments [53,107,125], glyphs [32], geographic layout [21,68,78], text representation [65], graphs [51,34] and similar high-level techniques to support security analysts. While current techniques have been proven useful through anecdotal evidence and evaluations [10,13,67,29], we believe that they should be combined with low-level representations of network packets, including payloads, to create domain-specific highly interactive systems. To this end, we present both a novel low-level visualization technique that we call a binary rainfall and a visualization system which allows users to semantically zoom [11] through eight representations of network traffic. We also explore the use of dynamic queries [122] and interactive encoding to enhance the performance of the system.

Some of the most promising network security visualization systems to date make excellent use of packet header data, particularly at the network and transport layers. While very useful in certain instances, these systems do not adequately visualize data at the application level and

subsequently miss payload based malicious activity. To address this issue, our binary rainfall technique allows users to compare payloads of 600-1000+ packets at one time.

While text has been the most common way to examine packet payloads, some research has been completed that uses visualization and other techniques. Axellson combined a Bayesian classifier with visualization to support analysis of HTTP payloads [5]. Signature based intrusion detection systems, such as Snort [96], excel at the pattern matching of packet payloads. Ethereal [35] is an exceptionally comprehensive protocol analyzer, which dissects packets and displays payloads in textual format. In the area of intrusion detection, Wang and Stolfo demonstrate a statistical approach based on the byte frequency of packet contents and return results in the form of byte frequency histograms [120]. In the area of file visualization, Yoo constructed self-organizing maps based on the executable content [128], abstracting away the underlying low-level binary structure. In the related field of binary analysis, the IDA Pro disassembler [27] and, to the best of our knowledge, all other reverse engineering tools, rely solely on the textual representation of binary objects. The common characteristic of these approaches is that they display the payload solely in textual format or abstract away the payload altogether and produce only higher-level information. Our graphical approach attempts to break the paradigm of the canonical hex and ASCII format used by these approaches, while at the same time, allowing the analyst to view low level details of packet headers and payloads.

To test the efficacy of the rainfall visualization, semantic zoom, dynamic queries and interactive encoding, we implemented a system that performs both live packet capture and forensic analysis using libpcap formatted files. We examined a variety of malicious and non-malicious network traffic including DEFCON Capture the Flag [94], United States Military Academy CyberDefense Exercise [116] and Honeynet Project datasets [56] as well as datasets

collected from a Botnet sinkhole created at Georgia Tech. These datasets included worm traffic, buffer overflows, network scans, trojans and other malicious activity. In addition, we examined several classes of typically non-malicious activity including SMTP, HTTP, Telnet, SSH, VoIP, FTP and SSL. From this experience, we present results and lessons learned.

To summarize, the primary contributions of this portion of our work are a novel visualization technique that allows comparison of hundreds of binary objects, experimental results from using the technique on malicious and non-malicious network traffic datasets, an application of the semantic zoom paradigm in the security visualization domain and lessons learned from the design and implementation of the underlying system.

User Needs Assessment and Human-centric Evaluation of Security Visualization

The preface to the Visualization for Computer Security (VizSec) 2005 proceedings contains a telling statement. The editors state that approximately 20% of the accepted papers incorporated user studies, but that they believed this was an important improvement. This is indicative of both the current state of security visualization research as well as for security systems in general. Despite the distinct shortage, there is useful representative work to help inform the design of our user study and experiments. As we reviewed these papers and others in the security evaluation literature, a consistent theme emerged. Quantitative evaluation requires a small number of tasks and straightforward metrics in order to accomplish useful results using a limited number of qualified and available study participants. As an example, three of these studies [39,121,9] used a small number of participants for key elements of their evaluation: 12, 43 and 6 respectively. Our assessment is echoed by research that indicates that only five individuals are required to find 80% of usability problems in a given system [79]. Another study [62] involved 1731 participants,

but this was a unique case involving phishing emails. Subjects in this instance were not informed of their participation in the experiment and were tricked, in large numbers, into participating via a misleading email.

While there has been some initial research that explores user requirements to guide the design of security visualization systems [10,13], formal studies of security analyst requirements are limited. Of those studies, there have been several good examples of requirements gathering from professional security analysts. Initial work was completed by Yurcik [129]. In this study, four National Center for Supercomputing Applications (NCSA) security staff members and several incident response team operators contributed security related operational interface requirements. The primary interface requirement being the need for an overall view of an entire network that provided situational awareness. Goodall conducted a series of semi-structured interviews, lasting one to two and a half hours, with nine individuals. Two members of this group were information security analysts and two were intrusion detection system developers. The remainder were network administrators. All participants had some intrusion detection experience. The results of this survey led to the development of a three-phase task model for intrusion detection tasks and insight into analyst collaboration [44]. Subsequently, the authors have supplemented the interviews with a focus group and prototype usability evaluation for a total of 16 participants. Their extended study led to the creation of an information visualization framework for intrusion detection [66]. D'Amico conducted a U.S. Government sponsored For Official Use Only (FOUO) cognitive task analysis of 41 information assurance analysts [26]. Some results from this work are publicly available [25] and have led to insight into how security visualization can be used to enhance situational awareness. Our work differs significantly from these studies in that we focus on the strengths and weaknesses of widely used intrusion detection

and packet level analysis tools. In particular, we examine the specific cognitive overload issues analysts face when using these tools as well as analyst's perception of the tool's susceptibility to the injection of malicious data. General information on the injection of malicious data to attack human operators is available [87], but is primarily theoretical in nature.

Human-centric evaluations of security visualization systems are quite sparse in the literature. Most are informal reviews conducted by a small number of potential users. Results are primarily anecdotal in nature. The remainder of existing studies are qualitative. Recent examples include Fink's evaluation of the Portall process level visualization tool [37] and Komlodi's evaluation of the Intrusion Detection Tool Kit (IDtk) [67]. While qualitative evaluations provide useful insights, quantitative evaluation is critical to effectively compare and contrast tools as they seek to meet user requirements. To the best of our knowledge, our work (see Chapter 6) is the first formal, quantitative evaluation of a security visualization system.

CHAPTER 3

A TAXONOMY AND FRAMEWORK FOR COUNTERING DENIAL OF INFORMATION ATTACKS

Denial of information is a critical problem, but current research is scattered and application specific. This chapter provides an overarching umbrella to better understand the problem space. The definitions, taxonomy and framework provide the clarity and structure required to begin holistic research in countering denial of information attacks. We can then place the use of information visualization within the larger scope of denial of information solutions as we move forward to the research we present in later chapters.

Terminology

The following are the primary objects and concepts from the framework we propose:

node – A machine producer and/or consumer of information. Each node has available finite amounts of processing, I/O and storage resources. In most cases, nodes are interconnected through communication channels.

communication channel – The medium used to transmit a message from transmitting node to receiving node. A communication channel has a finite ability to transmit information.

information object -- All individually addressable objects within the data universe. These objects fall into two categories: the information that you seek (signal) and information not sought

(noise). Malicious entities may design inappropriate information objects so that they appear as signal or design sought after information objects such that they appear as noise.

information universe – The set of all possible nodes, information objects and communication channels.

human -- A user of a node. The human interacts with the node using vision and hearing for input as well as motor skills and speech for output. Each human has available finite amounts of cognitive processing, perceptual capabilities and memory.

Denial of Information (DoI) - reduction of the ability of an information node or human to acquire desired information and as a result may be deceived or lack the available resources to complete a given task. When the attack is written to target just the computational resources of the node it is considered a traditional denial of service attack.

Denial of Information Taxonomy and Framework

This taxonomy and framework considers both the objects and actions associated with the denial of information domain. It is a result of consultations with domain experts who analyzed the design for completeness and applicability. In addition to domain experts, we observed users in denial of information scenarios. While it is difficult to prove the absolute validity and completeness of the taxonomy, we conducted 12 months of analysis at Slashdot.org and other meta-news websites searching for denial of information instances in any form. This research

allowed us to see patterns in the denial of information domain. Each new instance allowed us to iteratively improve and verify the taxonomy. To further refine the taxonomy we reviewed the proceedings from recent anti-spam conferences and several relevant knowledge management conferences and verified that each of the relevant objects and actions were reflected in the taxonomy. So while we cannot verify the absolute completeness of the taxonomy we do believe we have paid due diligence to make it comprehensive and accurate.

Overview

Objects in this framework fall into four main categories: information nodes, their interconnecting communication channels, humans and information objects. Actions include production, consumption, attacking and defending. Each information node can be a producer and/or consumer of information. Humans interact with information nodes to search for and, in some cases find, information that they seek. In this model, humans are also consumers and producers of information that they feed into or receive from information nodes. They work through information nodes to accomplish their tasks.

The goal of this framework is to characterize the components and characteristics in which denial of information attacks occur. This model framework recognizes three levels of granularity, but only addresses two: the coarse node level and the medium grain transaction level. It does not attempt to deal with the fine-grained semantic meaning of a given piece of information. As an example of this granularity consider the following:

- Node level - do you trust a given website.
- Transaction level - do you trust a given web page.

- Semantic level - do you trust a given fact on a web page.

We leave the integration of semantic level techniques for future work.

Categorization of Actions

Actions take the form of information attacks and defenses initiated by humans as well as the production or consumption of information. Attacks target the producer, consumer and/or the communication channel. Attacks exploit inherent vulnerabilities in the human or machine as they processes the data. Perspective is important to consider at this point. One individual's defense is another person's attack. Consider the case of a spammer who initiates a wave of emails in an attempt to sell a product. If some of those people use a technological defense such as email filtering or, perhaps, a legal defense such as a cease and desist lawsuit; what the recipients consider defenses are perceived as attack from the perspective of the spammer who is seeking information on potential customers.

Production

Producers are information nodes that provide information to other nodes. They may do this directly, by gathering information from other nodes, humans or sensors. The information they provide is within the spectrum of absolute accuracy to absolute inaccuracy. Their intent may be to provide accurate content, misinformation or a combination of the two. They may take countermeasures in an attempt to bypass the protection measures put in place by consumers. In some cases they may deliberately cloak the misinformation by surrounding it with a great deal of accurate information. This may also include modifying the information to bypass such protections. In the case of spam, subject lines may be manipulated in an attempt to penetrate the

consumer's filtering mechanisms. Producer nodes typically have a large number of outbound channels to consumers.

Consumption

Consumers are information nodes that receive information from producers. They may receive the information directly, as in the case of a web search engine query, or they may have the information pushed to them, as in the case of email. They may or may not have requested the information. Typically the goal of consumers is to acquire a certain amount of accurate information while filtering extraneous noise. In most cases, the more noise that penetrates their defensive mechanisms the greater the likelihood of a successful denial of information attack.

Attacks

Attacks may be intentional or unintentional, but always degrade the quality of information received by the human. This is manifest in a lower S/N. An attack manifests itself typically by a human giving instruction to an information node. Table 3.1 illustrates the following scenarios.

Table 3.1: Denial of Information Attack Scenarios

	Signal(S)	Noise(N)	S/N	Impact
Scenario #1	Low	Low	Parity	Marginal to good ability to find information.
Scenario #2	High	Low	Good	Good to excellent ability to find information.
Scenario #3	Low	High	Bad	DoI
Scenario #4	Very High	Very High	Parity	DoI, processing, I/O or storage capability exceeded (aka DoS)

An ideal web search by a consumer using a reputable search engine will return results where noise is low and signal is high (Scenario #2). In the case of a malicious node, the results might include pay-per-placement advertisements in response to the query, thereby increasing the noise (Scenario #3). The consumer may incorrectly formulate their query and receive only marginal results (Scenario #1) or through no malicious action at the producing node receive incorrect information (also Scenario #3). Attacks succeed by preventing the consumer from receiving the information they desire in the time window they need it or by transmitting noise that incorrectly appears to be valid information. This usually occurs when the information processing capability of the human is overloaded, but may also occur if the information processing capabilities of their automated information processing hardware and network cause the problem (Scenario #4). Both the attacker and the defender may take countermeasures in order to increase the likelihood of their success. The success of attacks is measured by the influence it has on the decision making of the target. In other words, how much change is there in what the target does or does not do. Table 3.2 shows these attacks in more detail.

Table 3.2: Denial of Information Attack Taxonomy (By Target)

Human	Processing	Cognition	Memory	Perceptual Buffers	Intentional	Magician's sleight of hand
					Unintentional	Web browser status bar displays information for too short a period of time
				Short Term	Intentional	Choosing a long password of random characters
					Unintentional	Forgetting a URL you were just told
				Long Term	Intentional	SPAM from "familiar" sounding source
					Unintentional	Remembering multiple passwords
			Cognitive Processing		Intentional	Using advanced technical jargon to exclude newcomers
					Unintentional	Inability to understand a web search engine's advanced query instruction
	Input	Vision			Intentional	Blinking Advertisements
					Unintentional	Small fonts
		Hearing			Intentional	Playing very loud rock music outside Manuel Noriega's home
					Unintentional	Background music at Website
	Output	Speech			Intentional	Speaking too softly to be eavesdropped on
					Unintentional	Inability to speak the language of the listener
		Motor			Intentional	Monitoring keystroke dynamics for authentication
					Unintentional	Cannot type fast enough
Machine	Processing				Intentional	An attacker floods an operating system with inbound packets, causing some to be dropped
					Unintentional	Software cannot perform decrypt function fast enough and loses connection
	Input				Intentional	Smurf attack
					Unintentional	Slashdot effect
	Output				Intentional	Encrypting output so as not to be eavesdropped on
					Unintentional	Inability to communicate with a required protocol
	Storage		Primary Storage (ROM/RAM)		Intentional	Corrupting the ROM BIOS of a system
					Unintentional	Running too many large applications concurrently
			Secondary Storage (Hard Drive)		Intentional	Sending network traffic to flood an intrusion detection system's logs
					Unintentional	Exceeding an email storage quota

Defenses

Defensive countermeasures may be utilized by both the attacking and defending nodes to increase or reduce the likelihood of a successful denial of information attack. A defender uses legal, regulatory, moral, cultural, organizational, financial, technological and perhaps even violent countermeasures to reduce noise and/or increase the signal in their information environment. An attacker may use the same measures to increase the likelihood of a successful denial of information attack. Table 3.3 lists a taxonomy of such countermeasures.

Table 3.3: Denial of Information Defense Taxonomy (Big Picture)

		Example
Legal	Lawsuits	RIAA sues small-scale MP3 distributors
	New Laws	New York State No-Call Database
Regulatory	Government Regulation	FCC approves media consolidation
Moral	PR Campaign	Movie industry campaign against movie piracy
	Code of Ethics	A state Bar Association imposes Code of Ethics upon its members
Cultural	Communities	Exclusion from the group
Organizational	Topic counter-DoI groups	Anti-spam consortium founded
Financial	Increasing cost of DoI operations	Cost of postage increases
Violence	Violence against DoI perpetrators	Sending threatening messages to telemarketing firm
Technology	(see Table 3.4)	

Because this chapter focuses on technical countermeasures, Table 3.4 goes into much greater detail in this area. Each technique attempts to increase the signal to noise ratio and thereby reduce the likelihood of a successful denial of information attack.

Filtering – Filtering is a widely employed technique that removes noise from the environment. Current examples include keyword filtering, Bayesian filtering and collaborative filtering. Another example is human in the loop systems that attempt to increase the quality of information by requiring a human representative to review the information before it is entered into a database.

Resource Consumption – Resource consumption based defenses force the potential attacker to pay a given amount of a finite resource, typically money, processing or time to reduce the effectiveness of their attack. Adaptive and agile systems allow humans to better acquire information by applying only the appropriate amount of computing

resources required for a task. This savings frees resources that can be used for other information countermeasures. Agents shift information-processing requirements from the human to information technology allowing the human to focus on other tasks.

Meta-Information – Meta-Information countermeasures attempt to match externally computed information with a given set of data. Examples include data quality measurements and trust metrics.

Trusted Computing – Trusted computing attempts to ensure the integrity of an information node. If a system is compromised then any information it provides is suspect.

Data Fusion – Data fusion attempts to reduce the amount of data presented to a user by merging data from disparate sources in a way that produces useful information and lessens the cognitive load.

Database Keys/Indices – This technique places emphasis on high-quality database keys and indices. An example is a web search engine that will ban a website that attempts to use inappropriate keywords in order to gain better placement.

Online Communities – Individuals join online communities to interact on topics of mutual interest. These communities can apply techniques such as excluding counter-productive members and enforcing cultural standards to provide highly relevant information to members of the group.

Source Evaluation – Source evaluation techniques attempt to identify authoritative sources of information as well as noise generating sources in order to provide higher quality information. This evaluation may include challenge-response techniques or a test to determine whether the consumer is a human or machine. This category also includes consideration of source anonymity. Anonymity of sources is a double-edged sword. In some cases it can lower the quality of the information if the source believes they will remain anonymous. In other cases anonymity will improve the quality of information by allowing individuals to speak freely.

Structuring Data – Improved structuring of data allows more efficient retrieval of relevant information. Technologies such as XML embed knowledge into the information and subsequently improve the encoding.

Restricted Connectivity – By restricting connectivity, organizations can prevent or reduce the opportunity for data corruption. As an example, an "air-gapped" wide area network that provides connectivity to only organizational machines and no open Internet access will reduce the threat of direct data corruption to insider threats.

Translating Data – Translations convert data into information that better meets a human's needs. XML provides a rich set of tools to support transformations. Other examples include middleware that converts between disparate formats and language translation systems such as BabelFish.com.

Human Computer Interface – By defending the human computer interface, a user can efficiently interact with their machine. An excellent example is information visualization which reduces the cognitive burden on the human by presenting data in a more efficient graphical representation. Humans can then make better decisions or discover new information.

Data Protection – Systems that detect changes in information or intrusions help build trust in the quality of information provided by a given source. Examples include the TripWire system that can detect changes in the files on an information system or the use of public key cryptography to protect the integrity of a given document. More recently, detection systems are used in conjunction with "self-healing" technologies to repair damage caused by an attack. The related topic of information preservation seeks to safeguard information that may degrade or change over time. An excellent example is the caching of web pages by Google or the Internet WayBack machine that stores snapshots of web pages over time.

Locating Data – This technique assists humans in initially finding information as well as locating information that they have already spent time and other resources to locate. This can be seen in web browsers that provide bookmarks and history functions.

Table 3.4: Denial of Information Defense Taxonomy (Technology)

Technological	Filtering	Collaborative Filtering	Slashdot.org
		Filtering Algorithms	Link analysis, proximity searches, page rank computation
		Human-in-the-Loop	Yahoo human reviewers
	Resource Consumption	Money	Charging fee for each email
		Time	Sending AI generated email responses to spammers
		Memory	Adding more RAM to a workstation
		Processing	Systems that effect a small computational charge to prevent abuse.
		Bandwidth	Data compression algorithms
		Resource allocation	Agents
			Adaptive/Agile Systems
	Meta-information	Data quality measurements	DOD Guidelines on Data Quality Management
		Trust metrics	Advogato.org
		Currency	Googlebot/WebCrawlers
	Data Fusion	Reduction and merging of data	Air Traffic Control, Weather Forecasting
	Online Communities	Exclusivity	Orkut.com
		Cultural standards of behavior	Slashdot.org
	Source Evaluation	Anonymous Input	The Freenet Project
		Community Input	Communities of trust and reputation systems
		Authoritative Input	Your college professor's list of recommended links
		Hardware/Software Trust	Trusted computing
		Authentication/Testing	Use of CAPTCHAs to determine if a consumer is human.
	Structuring Data	Keywords	Including keywords with academic papers
		Improved Encoding	XML
	Restricted Connectivity	Air gapped network	Power plant control network
		P2P Communities	Waste by WinAmp.com
	Translating Data	Converting data to more useful form	XML transforms
	Human Computer Interface	Presenting Data	Information Visualization
		Interface Design	Effective location of interface elements
	Data Protection	Integrity Protection	Tripwire
		Preservation	Historical Archives such as the Internet Way Back Machine, Google cache
	Locating Data	Keeping things found	Bookmarks, browsing histories, shortcut links
		Searching	Google
		Indices	Yahoo
		Naming	Providing unique names for information objects

Examples

We have developed a taxonomy and framework to better understand the objects, actions and countermeasures associated with denial of information attacks. Now that we have done so, let's look at two examples of its application.

Spam

Spam is a prime example of a denial of information attack. The goal of the human mail recipient is to receive emails (information objects) that contain content of interest (signal) and exclude others (noise). The goal of the spammer is to penetrate the countermeasures put in place by each recipient to promote their product, service or agenda. Spammers typically send blanket email broadcasts to hundreds of thousands, perhaps millions, of addresses. This process results in inboxes filled with irrelevant email. The human must spend time cycling through the OODA loop in order to find messages of interest.

For our example lets assume that a given human, Alice, has received 47 emails of which 6 are of actual interest. The remaining 41 emails constitute noise. Of these 41 emails, 35 are "obvious" spam and the remaining 6 are convincing enough that Alice will need to examine them more closely. Alice's strategy to review and process email is:

- Scan email headers
- Delete obvious Spam
- Read remaining emails and then decide to delete or keep

Given this strategy, Alice works with the mail client running on her personal computer (information node) to read her mail. Alice will cycle through the OODA loop 47 times in

the process of reading her mail. Each step in the loop (observe, orient, decide and act) requires Alice's time and other resources to execute. She will spend significantly more time analyzing the "convincing" spam than she will spend deleting the "obvious" spam. If Alice's wasted resources exceed those she has available for her current tasks, she has been subject to a successful denial of information attack.

Frustrated by the wasted time, Alice employs several countermeasures to help protect her. She attempts to deter spam by getting a new email address and sharing it with only her personal friends (restricting connectivity). She explicitly does not place it on the web (addressing/naming). She builds rules that examine her incoming mail and moves messages addressed to her old email address to a lower-priority folder that she scans only occasionally (filtering). She plans to delete the old address after several months when she feels confident that the new address has been disseminated to the right people (exclusivity/restricted connectivity). She also considers switching mail clients to a text-based program that runs quickly and has efficient keyboard shortcuts (interface design).

Alice is heartened to hear that several laws have been passed making spamming illegal and that metered mail is being explored to make spamming less attractive as a business (financial/legal). She is concerned that the charge, albeit small, for email being considered may lead to unwelcome charges.

In this scenario, spammers are not static enemies. They employ counter-countermeasures of their own to keep up with their target's defenses, a kind of cold-war one-upmanship. They attempt to bypass filters by using compelling, but innocuous subject lines, spoofed return addresses and names as well as creative ways to construct their messages that are easy for humans to understand, but difficult for a computer to analyze.

Meta-news Website

Groups of individuals and organizations are also affected by denial of information attacks. This can be seen by examining the meta-news website Slashdot.org. Users of this site submit short articles for viewing and comment. If left unprotected from denial of information attacks, the quality of information available on the site would degrade quickly. In comparison, consider the low quality of information available on unprotected Usenet news groups and unmoderated mailing lists. Slashdot employs a variety of denial of information countermeasures to great effect. All story submissions are reviewed by Slashdot staff (human in the loop) for relevance and interesting content. Only the best are selected for publication. Once published, users comment on the story. A randomly selected subset of users have the ability to rate the quality of each comment (collaborative filtering). Comments submitted by registered users (authentication) receive higher initial ratings than those submitted by anonymous users (anonymous input). Readers may easily filter stories at any rating threshold from -1 very poor quality to 5 excellent quality. (data quality metrics, filtering algorithms, interface design). Standards of acceptable usage of the system have evolved over time and contribute to the unique culture and quality of the site. (cultural standards of behavior)

Long after a story is published, it is still available (preservation) and can be found via a search mechanism on the website (searching) or with a URL which links directly to the story (keeping found things found).

As we have seen, denial of information is a large problem with a wide range of potential solutions. The remainder of this dissertation will explore the use of one such solution, information visualization, to counter denial of information when dealing with computer

security related data. The next chapter begins this exploration by closely examining the specific denial of information threats to information visualization systems.

CHAPTER 4

INFORMATION VISUALIZATION THREAT MODEL

Information visualization systems are a powerful tool for countering denial of information attacks, but to do so they must be designed with security in mind. Such systems are vulnerable to attack, either from malicious entities attempting to overwhelm, mislead or distract the human viewer or from non-malicious entities that accomplish the same result by accident. Some might believe that today's systems are not potential targets for attack. Clearly there are many domains where security is of minimal importance, but increasingly information visualization systems are being used to support critical decision making. For example, intelligence analysis, law enforcement, network security and business decision-support systems exist in an adversarial environment where it is likely that malicious entities are actively attempting to manipulate human end users. We believe that there is a clear threat today and there will be a growing problem into the foreseeable future. For information visualization systems to maintain relevance security must be considered. Information visualization systems inherently have the human tightly coupled in the system loop. In most cases, the human is the decision maker who will act upon (or not act upon) the information presented and, as a result, the human is a high-payoff and likely target. Any point in the information visualization system may be attacked, from data collection to processing to final visualization, in order to impact human interpretation. A "minor" compromise of a single bit may have significant impact on the human (consider a change in the foreground color of a scatter plot to the background color). Major compromises may have far greater impact. Our primary goal is

to identify these threats and vulnerabilities, as well as develop principles to counter or mitigate these attacks. By identifying the threats and weaknesses of their system, designers can make appropriate decisions to mitigate these vulnerabilities.

To see a sample attack in action, consider a visual intrusion detection system designed to supplement classical anomaly-based and signature-based intrusion detection systems. Such systems are typically co-located with a firewall at the border between the internal institutional network and the public Internet. This vantage point allows the system to observe and collect selected data from network traffic at entry and egress from the internal network. Our example system collects header data from network traffic and visualizes it in real-time. In particular, it captures the source and destination addresses of communicating network nodes, network protocols in use, source and destination ports (used for process to process communication across an Internet Protocol (IP) network, e.g. port 80 for a web server) as well as calculates a timestamp for each record. An adversary may easily inject arbitrary data into the visualization system, intermingled with legitimate users' traffic, due to weaknesses in current networking protocols. In our example, the adversary knows the system operator on the night shift is red-green colorblind. They also know that the default settings on the visualization system map the very common (99+% of traffic) Transmission Control Protocol (TCP) to green, the User Datagram Protocol (UDP) to blue and the Internet Control Management (ICMP) protocol to red. In addition, the attacker knows that the target node has serious ICMP and UDP vulnerabilities. The attacker waits until late in the operator's shift and launches an ICMP based attack. The already tired operator does not notice the red packet amidst the much greater noise of green packets. In this case, the attacker took advantage of the visualization system's color mapping to target a specific user, but many other techniques could have been used.

We will describe and illustrate these attacks in later sections.

To help combat usability attacks against visualization systems this chapter includes several novel contributions: a framework for information visualization system security analysis, a taxonomy of malicious attacks as well as technology independent principles for designing information visualization systems that will resist attack. We illustrate and validate these contributions with results from the design, implementation and real-world use of our security visualization system.

Information visualization systems are potentially vulnerable to a wide spectrum of attacks ranging from overt to subtle. An obvious attack is to simply corrupt the data. Akin to a denial of service (DoS) attack, an attack of this nature is likely to be immediately noticed by human users. While significant, in this chapter we are concerned with the more subtle denial of information attack. Denial of information (DoI) attacks target the human by exceeding their perceptual, cognitive and motor capabilities. They reduce the ability of a human to acquire desired information. Even if a traditional DoS attack against a machine is not possible, the human utilizing the machine to process information may still succumb to a DoI attack. Typically much more subtle (and potentially much more dangerous), DoI attacks can actively alter the decision making of human visualization system users without their knowledge. More specifically, for any visualization system, if an attacker can inject data into the dataset being visualized, or otherwise alter the dataflow, there exists the potential to exploit vulnerabilities in the human or the machine system. This exploitation can be used to accomplish some or all of the following high-level goals (inspired by well-established military information operations doctrine [3]):

- Mask a change in objects or actions that the system user has observed.

- Block the system user's perception and/or identification of objects or actions being introduced into the visualization system.
- Reinforce the system user's preconceived beliefs.
- Distract the system user's attention from other activities.
- Overload the visualization system or user's data collection and analytical capabilities.
- Create the illusion of strength where weakness exists.
- Create the illusion of weakness where strength exists.
- Accustom the system user to particular patterns of behavior that are exploitable at the time of the malicious entities choosing.
- Confuse the system user's expectations about an object's attributes and actions. In particular, to effect surprise in these areas.
- Reduce the system user's ability gain situational awareness and effectively make decisions.

To accomplish these goals, we make a key assumption: malicious entities may insert data into the dataset being visualized as well as deny access to, corrupt or alter the timeliness of data generated and communicated by networked data sources. We believe these assumptions to be reasonable. Many visualization systems gather information from potentially untrustworthy sources (such as unauthenticated Internet users or physically insecure sensors). In addition, data integrity and data availability are likewise susceptible to manipulation both in storage and in transit. Current cryptographic techniques can, if properly implemented, protect the integrity of data, but cannot guarantee availability. Consider that a small network device in the path of data flow can slow down or speed up transmission of sensor data despite cryptographic protection. Even more simply, a sensor

could be unplugged at tactically important times. Given these assumptions, it is important to note that we will not concentrate on the more traditional, non-malicious problems associated with designing information visualization systems as we believe they are currently being addressed. In most cases the problem of DoI attacks will remain even if these issues are addressed. Nor will we address general system attacks designed to broadly compromise as this is well addressed by the systems security community.

We argue that the ultimate goal of attacks against information visualization systems is to overload and deceive the human end users and force them to make incorrect conclusions and to take incorrect actions -- the exact antithesis of the goal of most information visualization system designers. This manipulation can be accomplished in a variety of ways, but ultimately these attacks corrupt data or alter dataflow in some way. They may occur quickly or over a long period at a barely perceptible, low level. The manipulation may take place at data generation, in transit over a communication network, at rest on a data storage device or during processing by a visualization engine. Attacks may be aggressive and essentially deny productive use of the system or may be subtle and covertly mislead. Either way, the result of an attack is an inaccurate picture as interpreted by the human end user. We have extensively reviewed these attacks and, for purposes of this chapter, we will place emphasis on the more subtle attacks, but will also provide coverage of interesting more aggressive attacks. Aggressive attacks are almost certain to be noticed, but subtle attacks are more insidious and may be overlooked for an extended period of time. As a result, the negative impact of these attacks may be far greater.

The threats to information visualization systems are legion. Attackers may range from trusted internal users to external competitors and be motivated by competitive advantage,

curiosity, intelligence gathering, notoriety, intellectual challenge or financial gain. To counter these attackers we argue that the only path to secure systems is via a thorough understanding of the possible threats and countermeasures. An effective technique to help secure systems is to conduct a threat analysis. Typically, this analysis includes the following elements: identifying assets you wish to protect, brainstorming known threats to the system, ranking the threats by severity, choosing how to respond to threats and choosing techniques and technologies (if any) to mitigate the threats [57]. We will include these elements during the course of the chapter.

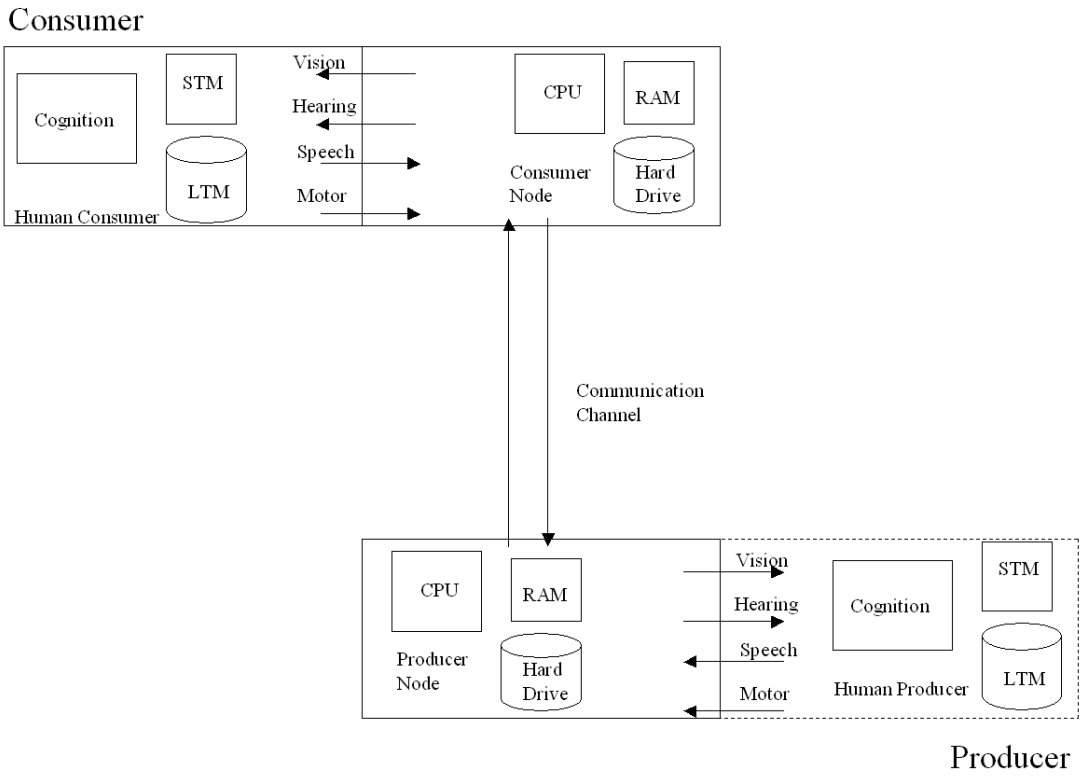


Figure 4.1. Generic producer-consumer information visualization system. Attacks influence any component, but the human end-user is the ultimate target.

SYSTEM MODEL

To best understand how attackers can accomplish the high-level goals presented in the previous section and to analyze how malicious visualizations manifest, we developed a generic producer-consumer information visualization system using a holistic systems approach (Figure 4.1). This architectural overview is useful for identifying assets by decomposing visualization systems and applications. The results can then be used to identify and prioritize the implementation of countermeasures.

The consumer is a combination of a human and machine. The machine presents the

information to the human using a visualization method that relies on one of the human's senses (typically vision). The human interacts with the interface using motor and speech commands and will draw conclusions based upon the information presented. The producer is the source of the data that will be visualized. In some cases, the producer will include a human who interacts with an information system to produce all or a portion of the data that will ultimately be visualized. In other cases, the producer will consist of only an information system that generates the data. No human is directly involved in data production (e.g. a sensor network). The producer may be co-located with the consumer, but it is more likely that the producer will need to communicate the data to the consumer via a communication channel.

Each human and machine component processes data using subcomponents with finite resources. Attacks can target any of these resources. For the human, we chose to model these resources based on the Model Human Processor (MHP) architecture: short term memory, long-term memory, cognition as well as perception and motor processing [16]. For each machine, we used the common information systems model of machine resources: processing, short-term storage (RAM) and long-term storage (typically optical or magnetic storage media). The human and its associated information system interact through the classic human-computer interaction boundary. The human utilizes vision, hearing, speech and motor actions to interact with the information system. Other senses (e.g. touch and smell) are not shown in the model, due to the limited availability of effective interface technologies. The information system provides related input/output devices that support each of these human capabilities (e.g. CRT, speakers/sound sub-system, microphone, keyboard and mouse).

Target					Attacks	Possible Countermeasures
Human	Processing	Cognition	Memory	Perceptual Buffers	Force desired colors to be used Force smaller font	Review annotation algorithms Limit range of colors, sizes allowed Review preattentive literature for best interface objects
				Short Term	Display updates to rapidly	Compensate with buffers in the visualization system
				Long Term	Aggregation hides important detail Scaling lacks detailed enough resolution Attack paging of visualizations	Lack of long term overviews Background images of historical data Use of paged and side-by-side images and overlays Create smart book of visual signatures
			Cognitive Processing		Degrade trust in system Attack when human is not watching Cry wolf Visualization software causes poor conceptual model	Display visualization's source data Create visual log files Ambient visualization
	Input	Vision			Causing occlusion of visual elements to conceal or manipulate visual presentation Inserting random noise into visualization Force less detailed scaling Occlusion of visualization elements Color choices impact color blind user	Develop alternative visualizations and views of data Include customizable filters Provide multiple coordinated views of data Choose smart default settings
	Output	Motor			Cause alert which forces user motor response (e.g. clicking an OK button) Force the user to scroll UI requires unnecessary actions	Review improved triggering mechanisms Explore alterative interface designs

Table 4.1: Denial of information attack taxonomy against information visualization systems.

INFORMATION VISUALIZATION ATTACK TAXONOMY

While attacks may range from overt to subtle, they share several common properties: they attempt to influence how you visualize, what you visualize or when you visualize. To this end, we present a taxonomy of attacks that follows the flow of information from human consumption back to initial data generation. We have developed a comprehensive taxonomy of attacks, but for purposes of this chapter, we provide a representative overview of the taxonomy and illustrative examples to highlight the vulnerabilities and surprisingly effective exploits of traditional information visualization systems. We have chosen to follow the information flow from the human back towards data generation, believing that this is an intuitive and natural way to illustrate an interesting spectrum of attacks. We will use the components along the path (see Table 4.1) to illustrate how and when attacks may manifest. Attacks may influence any component, but the human end-user is the ultimate target.

ATTACKING THE HUMAN

Humans are vulnerable targets with finite resources to perceive, interpret and act upon information. Attackers consume these resources through information visualization systems by altering the accuracy, completeness, consistency and timeliness of information flows. By focusing on human limitations these alterations create incomplete, ambiguous or incorrect visualizations that result in frustrated analysis, reasoning and decision-making. These malicious visualizations increase complexity, alter or destroy patterns, distort sequences and disrupt exploratory tasks which in turn may cause confusion, disorientation, disbelief, distraction or lack of trust. While not necessary, the effectiveness of attacks can be enhanced by specifically targeting individuals and their

unique set of weaknesses and predispositions (consider our colorblind user from earlier in the chapter). The following sections examine attacks against the human using a slightly streamlined version of the Model Human Processor (MHP) model of cognitive processing, memory, vision and motor resources.

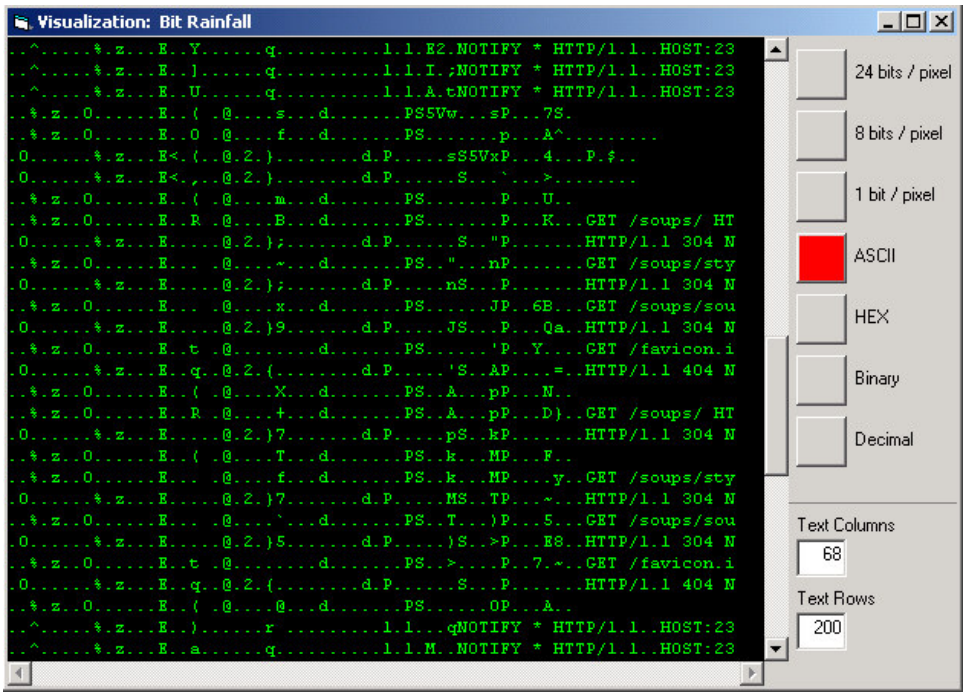


Figure 4.2. Semantic zoom visualization of network traffic.

Attacking Human Memory

Humans possess a limited ability to remember information over short and long periods of time. Arguably, humans can remember 7 +/- 2 "chunks" over a short period [76]. Regardless of the exact number, the human has a finite capability to retain and recall information. By exploiting this limitation an attacker can greatly increase their likelihood of success. These attacks may manifest themselves gradually such that the user fails to see the pattern. Alternatively attacks may target the users ability to recall legitimate activity to the detail required to detect malicious activity. Figure 4.2 illustrates this

limitation. This system, designed by the authors, attempts to provide a semantic zooming capability [11] for network traffic by allowing the user to view network information at variety of different scales from course grain overviews to high-resolution detail. The user selects the level of resolution using the scale on the right of the interface. Despite this attempt at allowing users to compare network traffic, it suffers from limitations of human memory. In our tests using the current configuration, users simply could not retain the context from one level to the next. Attackers could clearly exploit this weakness. To the best of our knowledge, no security visualization systems directly support the ability to closely compare images for subtle differences required to detect this class of attack. While Unix systems can use the diff command to compare text files, there is no equivalent visual diff. Likewise, there are no security visualization systems that allow users to seamlessly compare images in a side-by-side manner frustrating effective comparison.

Attacking Cognitive Processing

Cognitive processing deals with how humans interpret information. By exploiting weaknesses in this processing, an attacker can mislead the human and obscure or camouflage attacks as well as lead users to incorrect conclusions, perhaps even frustrating the users to the point they abandon use of the system altogether. Attacks can target attention, perception of time, decision-making, pattern recognition and perception of color and shape. Attackers may increase cognitive complexity, add spurious packets to eliminate suspicious outliers or demand the attention of the user. The following sections illustrate representative cognitive processing attacks against human attention and perception.

Attention

By their nature, information visualization systems require human attention. Depending on the design of the visualization and user interface the system may likely be tightly coupled with the user. It is impossible for a user to maintain 100% focus on their visualization system for long periods of time. Even a distraction lasting a few seconds can cause a user to miss key information. Alternatively, the attacker may overwhelm the user by demanding too much attention.

"Cry Wolf" Attack: From the classic children's story, an attacker can trigger activity, which in a normal scenario would require user attention. As a result, if the system "cries wolf" enough times the operator will begin to lose trust and may disable the system altogether. As an example, an attacker may subvert the snort intrusion detection system by creating packets that trigger alerts [87]. Snort alerts the user when it detects a signature in network activity that matches a known attack in its database. The snot tool is specifically designed to attack users through snort [95]. Utilizing snort's database of signatures, snot can generate network traffic that matches alert rules. Using snot, an attacker can trigger an alert at will.

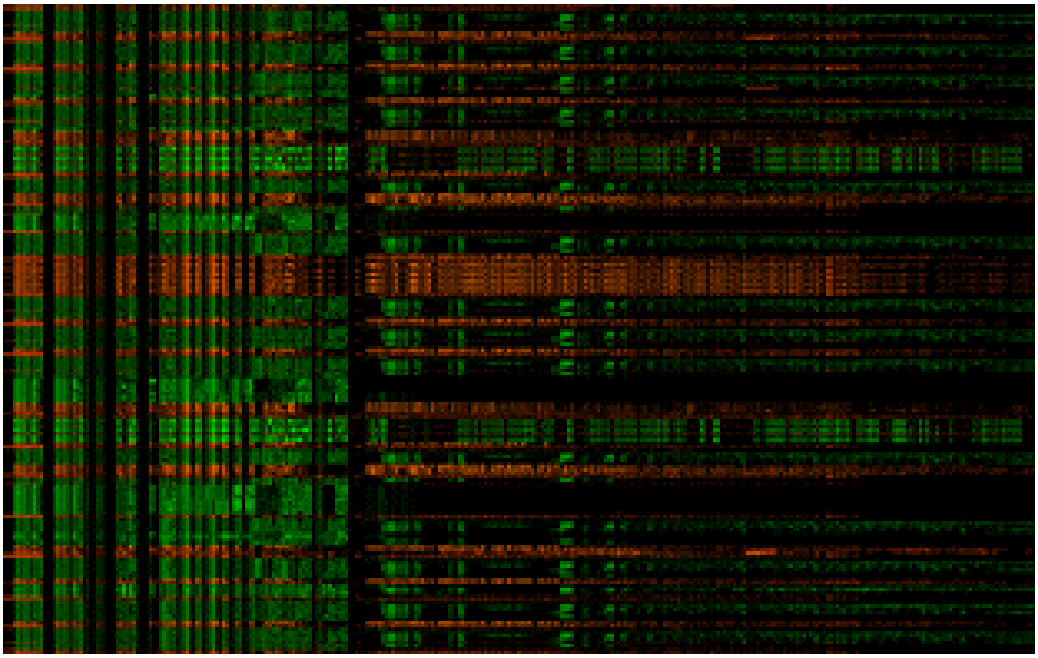


Figure 4.3: Binary rainfall visualization of network packets. (One packet per row)

Displacement Attack: Displacement attacks occur in visualizations where incoming data visually displaces older information. These visualizations are particularly susceptible to the limitations of human attention. Figure 4.3 is a network monitoring and intrusion detection visualization from the RUMINT system that displays network traffic in a scrolling display. The bits of packets are plotted on the horizontal axis. As each packet arrives it is plotted one pixel lower on the vertical axis. When the display reaches the bottom of the display window, it begins plotting at the top of the display, overwriting previous contents. During the past year we have used this system in two operational settings. The first was with the Georgia Tech Honeynet and the second was with a dedicated commercial Internet Service Provider (ISP) residential connection. In both

instances, the network connection is not used for any legitimate traffic thus only malicious activity is seen. Network packets typically arrive in small groups averaging one to five minutes per packet. Scrolling in these instances is typically not a problem, as approximately 24 hours of traffic can be seen before older information is overwritten (although we have seen spikes in traffic where network activity has been significantly greater). To test the time required for an attacker to scroll information off the page we conducted several experiments and found that it required only 2-3 seconds to overwrite information on one of our research machines (AMD 2500+, Windows XP, 1GB RAM, 100MB Ethernet). It is important to note that the theoretical limit based on network bandwidth alone is on the order of ten-thousandths of a second. We believe that a small lapse in attention on the order of seconds, even by a dedicated observer, is a reasonable possibility that an attacker may exploit to destroy traces of malicious activity.

Attacking Visual Perception

Information visualization systems, and the great majority of interactive computing applications, rely heavily upon the human's perceptual capabilities. Visual perception is the processing of input stimuli based upon reflected wavelengths of light from which our brain constructs internal representations of the outside world [20]. By taking advantage of the strengths and weaknesses of visual perception, an attacker can alter this internal representation. Consider the optical illusions from classic psychology. Given the same input image, different subjects might interpret the picture differently. In other examples, subjects are unable to correctly judge spatial relationships. See the work by Bach for 52 online examples [6]. Examples of other known weakness include a blind spot in the visual field, motion induced blindness [14] and a limited ability to discriminate between

colors. Even adaptations, which can be considered strengths in some instances, become weaknesses when manipulated by an adversary, such as preattentive processing [55] and afterimages caused by light/dark adaptation [77]. Beyond simple manipulation, even more aggressive attacks are possible. Small delays in virtual reality visualization systems can cause queasiness [52] and fifteen to twenty frames per second of certain images can trigger photosensitive epilepsy.

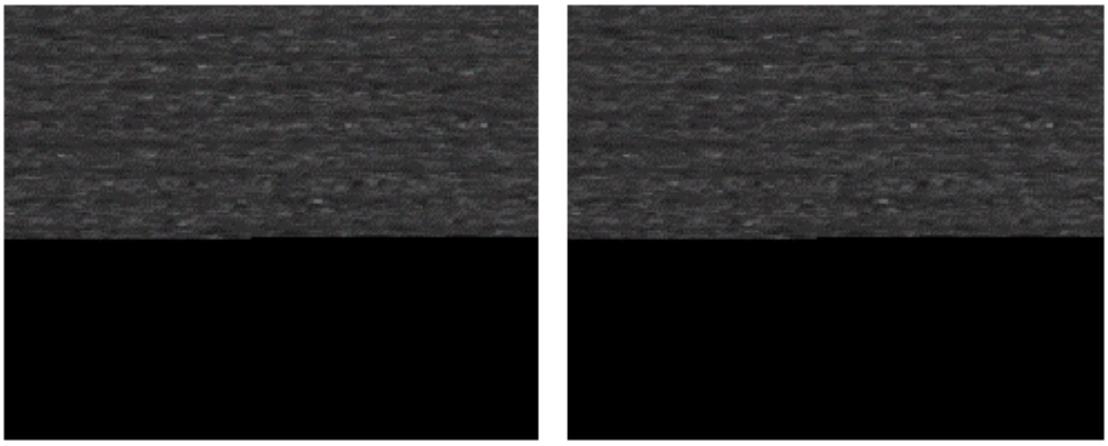


Figure 4.4: Binary visualization of two JPEG files. The left image is unaltered and the right image contains a steganographic message. Bytes from files are mapped to 256-level grayscale pixels.

Color Mapping Attack: The color mapping attack targets the use of color in visualizations. Humans have a limited ability to discriminate between colors, on the order of 300,000 colors [77]. Not all of these colors are interpreted as equivalent values, some are given heavier weight or draw more attention than others, and because color ranges are not uniform, normalization is used to help counteract the effect. See the work of Rogowitz and Treinish for an excellent discussion [89]. Most computing systems can present far more colors than a human can discern, 2^24 possible colors is typical on

today's hardware. Depending on the visualization system in use, features of the data are mapped, in a variety of ways, to colors used in the display. Limited display colors allow an attacker to hide activity due to aggregation. Large numbers of colors exceed or degrade the ability of humans to glean appropriate insights. It is due to these system presentation and human interpretation gaps that users are vulnerable, particularly when the system provides only a limited ability to customize colors for given tasks. Figure 4.4 comes from an analysis system we created to visualize executable files. It maps byte values from binary files to 256-level gray-scale pixels. In this example, the figure shows the file structure of two jpeg files. The left image is unaltered and the right image contains a steganographic message. Despite our ability to distinguish hundreds of thousands of colors, in our experiments, users were unable to find the modified bits. For future work we plan to pursue a visual diff tool, but the fact remains that for even a small number of colors, humans have difficulty in detecting differences. This weakness allows malicious entities to operate below the detectable threshold. Even the addition of a color legend is of little value. In a separate experiment we plotted network packets on a scatter plot using a commercial system. Even with only 100 different colors mapped to packet features (colors were chosen by the system) and a color legend, users took considerable time to match the respective color to the appropriate value. In another experiment, using the same commercial system and a scatter plot, we plotted 1,358 different network packets. We exceeded the number of categorical colors the system could provide and were forced to use a continuous scale. In this mode, no legend was provided. It proved impossible to identify the feature value from the color.

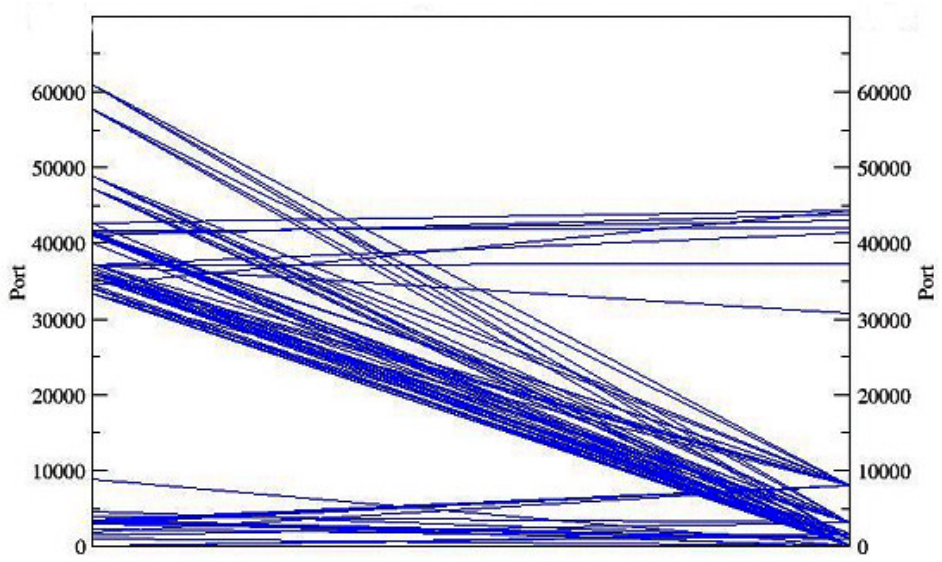


Figure 4.5: Autoscale and motor resources attack example (overview)

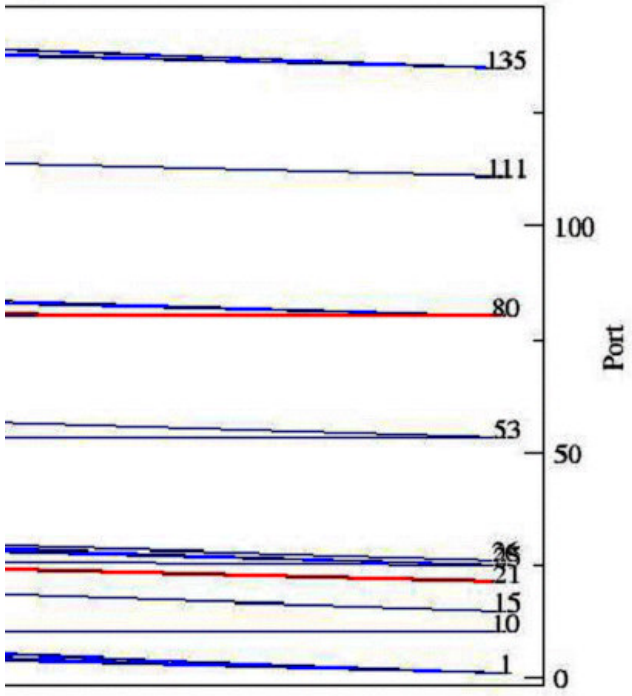


Figure 4.6: Autoscale and motor resources attack example. Note the targeted network services, originally hidden from view.

Attacking Motor Resources

This class of attack attempts to consume time and increase frustration by forcing user motor actions. Attacks may be as simple as forcing paging across multiple screens. Consider the RUMINT system described in the displacement attack, but add a buffer that stores previous pages of images. As each screen is filled, the user must interact with the interface to observe previous activity. Another example is to force user thrashing by requiring constant swapping from detail to context and back. Figures 4.5 and 4.6 illustrate this attack. The dataset behind these figures comes from an unclassified attack/defend exercise, in which a National Security Agency red team attacked student-defended networks [116]. The user is presented with an overview of network activity in Figure 4.5, but to see the specific port-level the network activity in Figure 4.6 the user must zoom in and then back out to continue to monitor the overview. In this example the user would have to perform this operation ten times just to monitor the 1024 privileged ports on a Unix system.

Targeting Specific Humans

While the attacks described previously are significant, even more effective attacks are possible if the specific human user is known. With this knowledge, an adversary may craft an attack that specifically exploits their target's weaknesses. Vision, memory, reflexes, experience and intelligence vary greatly between individuals. Even partial knowledge of the specific end user gives the adversary an advantage; their attack may be markedly different for a 19-year-old male intern, a 37-year-old male disgruntled employee or a 58-year-old female veteran who has heavily corrected vision. We believe

that some degree of knowledge of the human user to be a reasonable assumption. A few casual questions asked at an after-hours happy hour frequented by company employees would likely gain useful information. A comprehensive discussion of all such attacks is beyond the scope of this chapter, but we will illustrate the vulnerability by examining photosensitive epilepsy. While this condition is relatively rare, it does illustrate the increased risk when the attacker can target specific people and their weaknesses. We argue that related attacks can be launched when age, gender and/or medical details are known about users.

Extreme Information Overload Attack (Photosensitive Epilepsy): Epilepsy has a lifetime prevalence of about 3% and approximately 2.3 million people in the United States have the condition. Of this population, a percentage has photosensitive epilepsy. People with photosensitive epilepsy are susceptible to seizures brought on by rapidly flickering or flashing displays. In the late 1990's, thousands of people were sickened with nausea and dizziness by a Japanese Pokemon cartoon. In addition, there were 685 cases of apparent epileptic seizures [54]. The risk extends beyond the viewing of shows on televisions and computer monitors. Video games have also induced seizures and many now carry warning labels. It is important to note that the video game and video industries have since taken other proactive measures to limit future incidents; reducing the overall incidence of the problem. For example, the Pokemon cartoons were reviewed, frame-by-frame, before rebroadcast in the United States [52]. An attacker would do the opposite. Research by Harding indicates that the larger the area of the retina covered with the flashing display, the greater the likelihood of a seizure. In particular, flashing at the rate of 15-20 times per second was shown to be most likely to induce a seizure; 96% of

people with photosensitive epilepsy were sensitive at this frequency. In addition to flashing, static patterns have induced seizures and the likelihood is dramatically increased when patterns are updated at the rate of 15-20 changes per second [54]. With the trend toward larger displays and higher resolution the situation is worsened. In our experiments we were able to generate network traffic that caused both static and updating patterns in our network visualization system that would possibly induce seizures in some photosensitive epileptics, but we did not proceed further due to safety concerns.

ATTACKING VISUALIZATION HARDWARE AND SOFTWARE

The attacker affects attacks against the human by influencing how information is visualized. As was the case for humans, the notion of specificity is important to consider. Many of the techniques described below are most effective when used against specific information visualization systems, but others are broadly applicable.

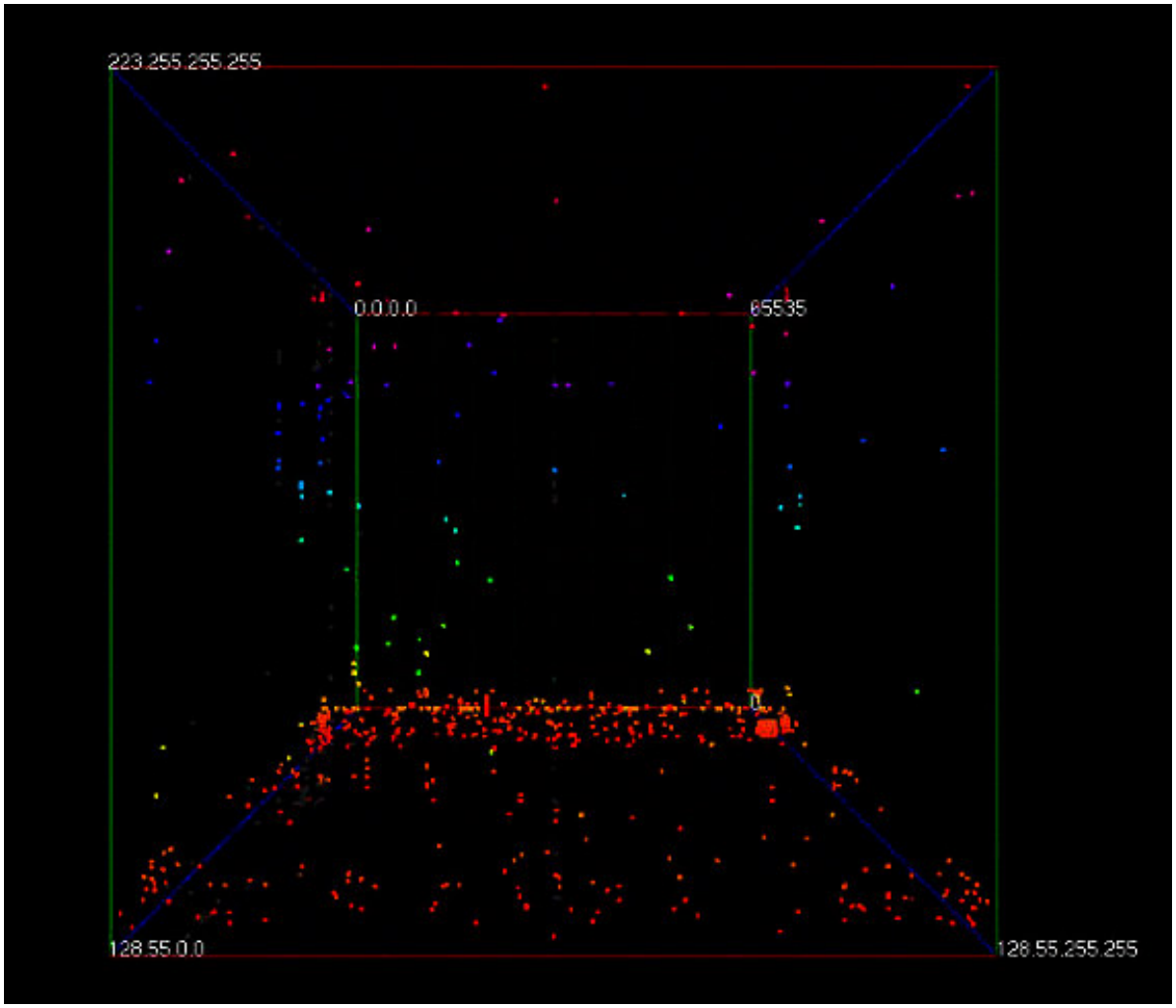


Figure 4.7: View of the "Spinning Cube of Potential Doom" a 3-D visualization of network traffic designed for security awareness. (round-off attack)

Processing Attacks

Processing attacks target the algorithms used to process and present the visualization. These algorithms range from simple graphic routines to advanced artificial intelligence or machine learning techniques. Attacks may be designed to increase computational complexity, e.g. creating a large number of objects such that the interface becomes sluggish or the visualization delays presentation of important information. Others may exploit intelligence embedded in the visualization system. Consider a generic spring layout algorithm. To be most effective, this algorithm relies upon the graph to reach a stable state. Carefully constructed packets could be used to force constant destabilization. Other attacks may take advantage of bugs in the code or the calculations in use, such as interpolation or round-off. To provide a concrete example of the efficacy of these classes of attack, the following section illustrates the round-off attack in detail.

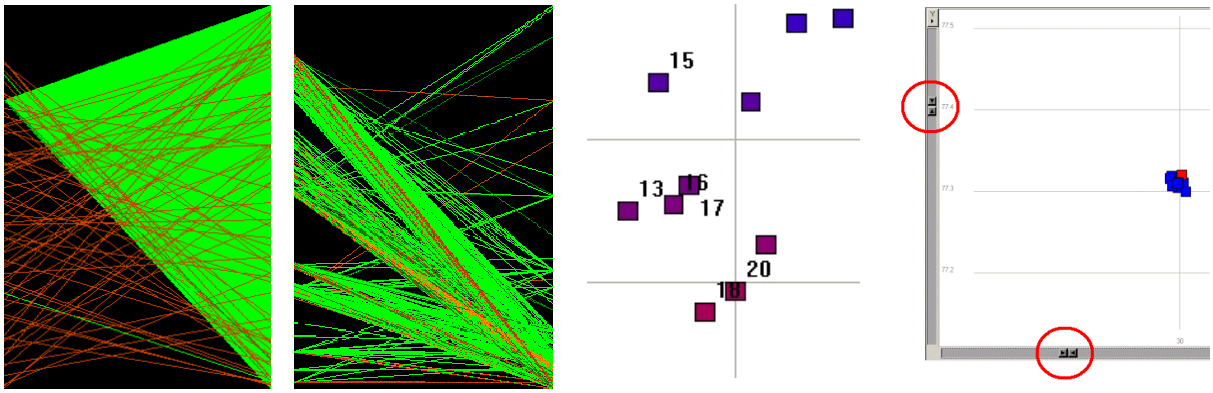
round-off attack: Consider the "spinning cube of potential doom" visualization system in Figure 4.7 [69]. Designed to provide insight into network attacks, it displays network traffic along three axes. The X-axis represents the destination IP addresses for a Class C network (65536 possible addresses), the Y-Axis displays destination ports from 0-65535 and the Z-axis displays source Internet addresses from 0.0.0.0 - 223.255.255.255 (no multicast). Assuming an approximate 1024 pixels for each axis. The X and Y axes round off 6 bits of information, leaving an opening for an attacker to operate within a space of 64 indistinguishable positions. More importantly, the Z axis rounds off approximately 22 bits of information, grouping source IP's into buckets of over 4 million each. Thus an adversary could attack 64 machines on 64 ports from over 4 million

source IP addresses and, due to round off, would only illuminate a single pixel. Note also that the visualization is also a target for a color mapping attack. It uses a "rainbow" color map representing TCP connection instances. Although a large number of colors are used, the actual color does not have "any meaning."

Attacking the Visualization

The heart of a visualization system are the visualizations it presents to the user. Closely intertwined with processing attacks, attacks against the visualization design will have an immediate effect on the user. Some visualizations were simply not designed to convey a certain type of activity, so an attacker may easily operate with impunity. In other cases, the design is such that a small amount of malicious data can destroy or reduce the effectiveness of the system. Designers are faced with large, potentially massive, datasets and limited screen real estate to present information and are forced to make design tradeoffs that can be exploited. The following are examples of such attacks.

autoscale attack: Many visualization systems use autoscaling algorithms to provide an optimal display of information. Typically the algorithms zoom the perspective outward to display the entire dataset. While this is convenient in many cases, an attacker can easily exploit this vulnerability. The image shown in Figure 4.5 was created by the xmgrace open source information visualization tool [47]. A small number of packets sent by the attackers to those ports above 40,000, forced the autoscaling algorithm to zoom outward, thereby obscuring significant detail (Figure 4.6).



(a) jamming **(b) occlusion** **(c) labeling** **(d) GUI widget**

Figure 4.8: Representative attacks against the visualization

jamming attack: The jamming attack is a simple attack, akin to a visual denial of service. By observing what aspects of the dataset are used to control the physical location of objects on the screen visual noise can be inserted to partially or completely corrupt the information display. As noise is inserted, insightful patterns, relationships, anomalies, outliers and trends will disappear. We produced multiple versions of this class of attack in our network visualization system by generating network packets with appropriate headers. Figure 4.8(a) is a parallel coordinate plot of TCP/UDP ports. The left axis shows the attacker's source ports and the right axis shows the target machine's ports (on a 0-65535 scale). The image shows two jamming attacks, both done using the packit packet creation tool [15]. The first attack generated 200 UDP packets (in orange) with random source and destination ports. The second attack (in green) generated 2000 TCP packets from a single source port, but random destination ports.

occlusion attack: Occlusion is a problem in many visualizations, particularly those in 3D, but any that plot new information over old are susceptible. An attacker can use this

frequent shortcoming to hide malicious activity. In the Figure 4.8(b), an attacker's malicious activity is hidden behind newer activity.

labeling attack: Typically visualizations provide the ability to label the image. Depending on the labeling algorithm in use, this fact can be exploited. One popular commercial visualization system defaults to only 20 labels. If the user does not change this setting a large number of objects will not be labeled, greatly complicating user interpretation. See Figure 4.8(c) for an example. At the other end of the spectrum, some labeling algorithms do not limit the number of labels used and, by injecting extra data, an attack could cause the display to be obscured.

GUI widget attack: User interfaces only provide a limited ability to interact with the dataset. An attacker can exploit this limitation and prevent users from detecting malicious activity despite their best attempts. Figure 4.8(d) shows a cluster of network activity; because of the large range of values in the overall dataset (not shown), the user is unable to zoom in any further. Any movement of the sliders will cause the entire cluster to move off the screen. Note the two red circles. Each circle shows a double-ended slider at the closest possible position.

Storage Attacks

From our research, storage attacks against information visualization systems can occur primarily in the form of classic denial of service. Denial of information and not denial of service is the focus of this chapter so we will touch only briefly on it here. Every information system has a finite amount of storage. By consuming all or most of this storage an attacker may subvert the intent of the visualization system. In the network security domain, a classic example is flooding the network with traffic, sometimes legitimate (also known as the slashdot effect) and sometimes malicious (trigger logging events to the point that the hard disk fills or malicious activity is overwritten). Variants include filling up the buffers of network interface cards such that packets are dropped or consuming RAM to the point that the operating system needs to page memory to disk (or even thrash). All of these attacks negatively impact performance and could crash or slow the system. While not strictly a storage attack, it is well documented that, in shared user systems, one user's applications can consume resources to the performance detriment of other users. Correctly designed interfaces operate within very strict timing parameters and a sluggish interface (or visualization) that quickly becomes difficult or unusable could quickly occur.

Attacking Data Generation and Communication

By definition, information visualization systems present data to the user in order to provide insight. If the accuracy, reliability, timeliness, completeness or currency is threatened then the entire system is at risk. Attacking data quality early in the system flow is a means to an end and not and end unto itself. The tainted data will ultimately flow upstream to the visualization system which, in turn, will alter the user's perception

and hence negatively impact task accomplishment. Recall that we do not consider data corruption attacks as we believe that they will be easily detected.

Attacking Data Generation

In our model, data can come from human and machine producers, both of which may prove unreliable despite the best intentions of the system designer. This notion is directly opposite to the common assumption that the "source must be good." While not the focus of this chapter, physical attacks are the most straightforward attack. The most basic is physical destruction or theft which causes a failure to record data. More subtly, an attacker may spuriously add, remove or compromise information producing nodes via physical access or network attack. Consider physically turning a sensor on and off (or cutting power) which results in selected subsets of data being recorded. Note that this could occur with more than one sensor and provides the attacker the ability to paint a customized and comprehensive picture of the information space. Beyond physical access, we consider attacks that allow an attacker to operate remotely.

sensor blindness attack: Network-based blindness attacks allow an attacker to remotely crash selected packet capture sensors on the network. As an example, virtually all Windows-based network sniffing programs use the WinPcap [124] packet capture library. Versions of the library have known vulnerabilities that will crash the sensor.

selective sensor blindness attack: Similar to the sensor blindness attack this variant exploits differing operating system implementations of the network processing stack to avoid detection. For example, one operating system may accept a packet with an

incorrect TCP checksum while another will silently ignore it. This inconsistency allows network intruders to operate without detection if the network sensor ignores the packet and a target machine accepts it. For more information see the work of Ptacek and Newsham [88].

spoofing source identity attack: Spoofing source identity is another common vulnerability, usually due to weak access controls or authentication, that allows users or network systems to appear as legitimate producers. In the network domain, it is trivially easy to spoof IP packets. The protocol offers no protection and an attacker may transmit packets with spoofed source addresses that appear to come from legitimate sources.

interface spoofing attack: Interface spoofing attacks have existed since the beginning of shared computing systems. Typically they are used to trick legitimate users into revealing sensitive information, such as passwords. In the context of this chapter, they can be used to trick legitimate users into submitting incorrect data to the visualization system. This technique can be seen when employing a variant of current phishing attacks. An attacker could send an email to a legitimate producer asking them to use a new website to submit information. Normal cues from the browser, such as the status bar, can be spoofed to prevent detection. See the work of Levy for more detail on this class of attack [70].

sampling rate attack: Sampling rate attacks exploit the periodicity of data collection. Due to the high rate of data flow observed by some sensors, by necessity, sample data at a constant or varying rate. This is typical in today's network visualization systems. Even

in near real time systems, a five minute sampling rate is common. By gaining an understanding of when data is sampled, an attacker can avoid detection.

poisoned data attack: Poisoned data attacks are carefully crafted to inject a small amount of malformed or incorrect data to disrupt collection or analysis. These vulnerabilities may exist due to a lack of input validation at the producer as well as the consumer's system. As we mentioned earlier, a single legal packet can have significant impact on the end user, as was seen in the autoscale attack. The same can be accomplished with a small amount of seemingly legal, but maliciously formed data. An excellent example, is the recent spate of image files that exploit vulnerabilities in image processing libraries. A single such image can crash a visualization application or provide privileged access to the attacker.

Attacking the Communication Channel

Communication channels connect the information producing nodes to the information visualization system. Long a subject of network security discussion, there are a large number of vulnerabilities in current networking protocols. If communication links are not secured with message confidentiality and integrity protection, an adversary may easily perform a "man in the middle" attack and arbitrarily alter packets between the producer and the information visualization system. Also, as we have discussed, the network layer (IP) provides virtually no protection from spoofing source identity and other tampering. Common transport layer protocols (TCP and UDP), similarly provide limited protection. UDP makes no attempt. TCP relies upon the three-way handshake and session establishment to prevent spurious packets. Handshaking and session

establishment provides only limited protection as an attacker can employ well-known TCP session hijacking techniques. Due to these weaknesses, an attacker can alter messages between producer and consumer at will, as well as observe all message traffic, unless some form of cryptographic protection is used. Even if a secured protocol is used, most will still be vulnerable to the following timing attack.

channel timing attack: By placing a properly configured network device in-line along the communication channel between the producer and the consumer, an attacker may affect a number of timing based attacks. The channel timing attack allows the capture and replay, both faster and slower than actual, of network traffic. By altering the timeliness of how and when data is presented to users, an attacker may reduce or increase data density or alter the distribution of data values causing a direct impact on the visualization and the human. Time-series data is particularly vulnerable to this class of attack.

PRINCIPLES FOR COUNTERING MALICIOUS VISUALIZATIONS

There is no panacea that will absolutely protect information visualization systems from attack, but there are important design principles and assumptions that will mitigate the risk. Recall that any information visualization system in which a trusted or untrusted adversary can inject or modify information places the end user at risk. As we conducted the research associated with this chapter we designed a variety of security information visualization systems and fielded them in operational settings. As a result of this experience we have learned a number of lessons. As you design or redesign systems of your own, we hope that you will consider these principles and assumptions. We believe

they will greatly reduce the likelihood of many classes of successful attack. In other instances, there is no clear-cut solution and the only countermeasure is awareness of the vulnerability.

From our experience, often the initial design of the system itself was at fault, leading to easily exploitable vulnerabilities such as the displacement attack. Others are more difficult to implement and potentially require detailed information about the system in use or the specific user. By using these principles and considering these assumptions during design, threats may be pruned or reduced and prudent design tradeoffs may be made. Ultimately, as information visualization systems are used for critical applications we must continue to explore how we can effectively deal with threats in order to make such systems more secure and relevant.

EDUCATE THE USER

The user is the ultimate target of attackers and the success or failure of an attack depends, in large part, upon their individual susceptibility. To counter many forms of attack, train users to be alert for manipulation, aware of their personal weaknesses and to take maximum advantage of system customization capabilities to counter these weaknesses. As a result, users will be better protected and resistant to attack. The intelligence community uses similar techniques to help prevent successful social engineering attacks through security awareness training.

ASSUME AN INTELLIGENT, WELL INFORMED ADVERSARY

Information visualization systems of any import will be targets of attack. Underestimate the attacker at your own risk [19]. To best protect a system we must assume an intelligent and well-informed adversary. The attacker may gain information through open-source (publicly available information) or through social engineering. Seemingly unimportant data may prove to be extremely valuable. As an example, such information as the time lunch was served and the location of the dining hall, both considered to be trivial pieces of information, possibly enhanced the attack on the USS Cole. It is not unrealistic to assume that an attacker knows the visualization tool in use. This assumption is strengthened in areas where a single tool dominates or there is a lack of diversity. In some cases, the attacker may possess the tool itself and the source code. This access allows an adversary full knowledge of it's operational characteristics and implementation vulnerabilities (buffer sizes, defaults, scaling algorithms, color mapping etc.) This assumption also applies to your users, the same social techniques that are used to gather technical information can also be used to gain insight into specific operators and environmental conditions. An intelligent and well-informed adversary will target your specific system through its weakest link, at the worst time with the weakest user at the controls. The best defense is to look at your system through the eyes of an attacker, predict their likely attack courses of action and consider what you can do to counter or frustrate their actions.

DESIGN THE SYSTEM TO PROTECT THE USER.

Assume the system, including the implementation and supporting information flow (from source to human consumption), will be attacked. Given this assumption, every creator of

a visualization system or technique should consider malicious applications and seek to create well thought out visualizations that are resistant to attack. At the time of creation, system designers do not necessarily know the full range of future use. Assume your system will be used for critical applications and attempt to predict second and third order effects.

Visualization systems typically have the human tightly coupled in the decision making loop. These systems require the limited resources of human attention and time, use them wisely. Even a small consumption of these resources by an adversary can cause unintended consequences on human decision-making. Customizable systems with intelligently chosen, attack resistant, defaults will help prevent overloading or deceiving the user, especially when combined with validated classical information visualization principles. If after your analysis, you cannot protect against a given class of attack before it reaches the user, at least assist the user in detecting one has taken place (detecting "wrongness").

PROTECT THE DATA GENERATION AND DATA FLOW.

An information visualization system is only as good as the data upon which it depends. Your ultimate goal is to improve data quality by increasing the good and reducing the bad, with emphasis on the most dangerous. It does not take much bad data to cause significant damage. In the network security domain, a single bad packet can provide root level access, waste hours of an operator's time due to a false snort alert or hide an attack due to an auto-scaling algorithm. In most instances, information visualization systems operate in environments in which an adversary can insert malicious data. Any source of data can be manipulated by a potentially malicious entity, including legitimate users,

machine producers and other trusted sources. Your data should be protected by well-validated techniques such as input and source validation and cryptography.

While it is beyond the scope of this chapter, designers should be aware of secure systems design best practices and threat modeling [101]. In particular, consider secure protocol development (confidentiality, authentication and integrity, in particular), appropriate use (and the limits) of cryptography, suitable security and usage policies, physical security, intrusion detection and input validation. In high-risk applications, physically closing the system to outsiders (air gapping) and the use of virtual machines to separate data and processing into logical groupings may be in order.

Information visualization is a powerful tool for countering denial of information attacks, but systems must be designed to resist attack in order to be most effective. We applied our principles for countering malicious visualizations as we designed and built our system as can be seen in the next chapter.

CHAPTER 5

RUMINT: A SECURITY VISUALIZATION SYSTEM

We addressed the problem of countering denial of information attacks by carefully crafting our visualization system to present data in insightful ways that tap into the high bandwidth visual recognition capability of human analysts. We began our work by surveying professional security operators to determine the limits of today's best systems and identify high payoff targets for improvement. (See Chapter 6 for the complete results.) Using these requirements to drive our design we created the RUMINT visualization system. While there is a wide range of potential security data sources, RUMINT is designed to provide detailed insights into packet level network traffic. We have deployed RUMINT in a variety of laboratory and operational settings for a total of two years to evaluate its effectiveness. During this period, we iteratively improved RUMINT's design and developed a general framework for the design of security visualization systems. In addition, we believe that the interactive, graphical techniques that we present will have broad application to other domains seeking to deal with problems of denial of information.

A Design Framework for Security Visualization Systems

In this section we propose a framework for the design of security visualization systems (Figure 5.1). While system frameworks are difficult to validate categorically, our proposal emerged from several years of focused research. During this period, we iteratively designed and implemented six security visualization systems and conducted an extensive survey of commercial, open source and research systems. From this review, we noted distinct similarities in the architecture and processing pipelines of many systems,

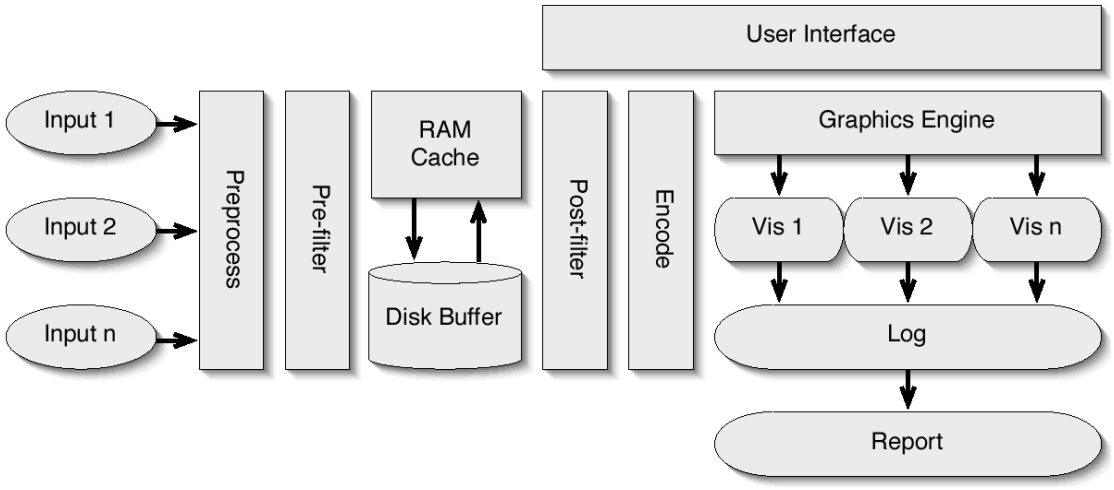


Figure 5.1: Framework for the design of security visualization systems. Security data flows through a series of intermediate processing, encoding and filtering steps before being visualized. After visualization, the results may be logged graphically and tailored reports may be created.

but were unable to find an underlying framework in the literature to inform our designs. We also noted that several systems advanced valuable, but rarely seen, components that we believe other researchers should consider. Finally, we received feedback from users on several key areas that were lacking from any of today's systems. By merging insights from all of these sources we have attempted to create a comprehensive framework. We believe that by better defining the components and processing sequence of security visualization systems, other designers will more rapidly be able to design and construct effective systems. In addition, by closely examining each individual component in isolation there lies great potential for future work and optimization. We also believe that the lessons learned, which are embedded in the framework, will assist researchers working in other domains, particularly those constructing interactive information visualization systems. The following sections describe the major components of the framework and illustrate its use by decomposing the Ethereal system. It is important to note that not all components of the framework must be implemented for a successful

system, but we believe designers should, at least, consider each stage.

inputs: Possible inputs to the system may take many forms across a broad spectrum of data quality, from unprocessed data to highly refined semantic information. For example, Snort performs signature matching against network traffic to provide specific alert information. Ethereal collects only raw capture data from network packets. Sources may include flows from security sensors, firewalls, intrusion detection systems, network servers, host-based security sub-systems and honeynets. Inputs are not constrained to these, typically passive, traditional sources. Additional semantic information may be infused into the visualization system by including active collection flows such as those provided by the nmap network mapping tool as well as more specialized tools, such as the p0f passive operating system fingerprinting tool. Single streams of security flows are used in most implemented systems; effective integration of multiple streams to support improved correlation remains an open problem. Timeliness of the data will range from real-time, near-real-time and historical information. It may be collected directly by the visualization system, received from external devices or pulled from intermediate databases.

preprocessing and pre-filtering: The data and information flows received by the system may or may not be in a format compatible with the system. In many instances they will need to be parsed and relevant information will need to be extracted. Ethereal does this comprehensively through the use of dissectors for over 700 different protocols. Pre-filtering allows users to select only the desired subset of records/fields to progress further up the processing pipeline in an effort to conserve system resources. Ethereal

implements this capability through the use of a capture filtering language.

system storage (RAM Cache & Disk Buffer): After preprocessing and pre-filtering, the data may be buffered. The buffer typically consists of a RAM cache and may include additional storage on disk for large data streams. Ethereal behaves in this manner. Such storage is optional and may be bypassed in instances where interactivity is at a minimum and state is not required.

post filtering and encoding: Filtering and encoding are logically intertwined. Before being passed to the graphics engine and subsequent visualization, the user makes choices based on what information they would like to view and how to display it. Ethereal uses a display filter language to filter data in the buffer and provides a coloring capability to encode additional information in the display.

graphics engine and visualizations: The graphics engine receives the remaining components of the data flows as well as encoding instructions and passes the information to the visualization displays. The visualizations display the information using a variety of information visualization techniques and may include any number of semantic windows on the data. Typically these visualizations are graphical in nature, but may exploit other senses such as sound and touch. Ethereal provides a three pane multiple coordinated display which includes an interactive textual list, tree based protocol decodes and the raw hex/ASCII representation of the selected packet. In the future, the graphics engine and visualization windows may include ties to machine processing modules to direct and conserve human attention.

logging and reporting: Visual logging and reporting are relatively unexplored aspects of security visualization systems, but our interviews with security analysts indicated that they are quite important, particularly the reporting task, for communicating results to other analysts, end users, customers and managers. Visual logging of security data includes automatically storing images and video clips of visualization activity in lieu of storing the underlying raw source data. Visual reporting exploits the strengths of visualization systems by allowing the analyst to work through slices of network traffic and, once an area of interest is determined, allows easy construction of a summary report that may include marked-up images, video clips, filtering parameters and analyst comments. Ethereal incorporates neither of these capabilities.

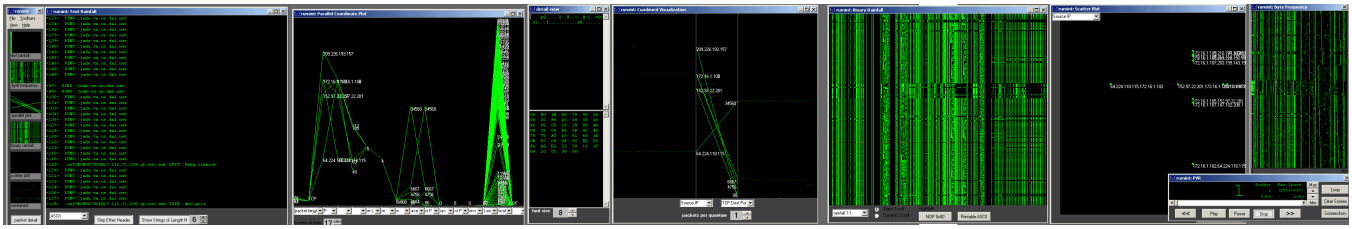


Figure 5.2: A panoramic view of the RUMINT packet visualization system. Each window is designed to provide specialized insight into network traffic. From left to right, the visualizations include: (a) thumbnail overview toolbar, (b) scrolling text display, (c) parallel coordinate plot of packet header fields, (d) detail view of packet contents, (e) glyph-based animation of network traffic, (f) binary rainfall visualization, (g) scatter plot, (h) PVR interface and (i) byte frequency display.

Design Overview

In order to best the design RUMINT to counter denial of information attacks we carefully considered the design principles we proposed during our threat analysis chapter: educate the user, assume an intelligent and well informed adversary, design the system to protect the user and protect the data generation and data flow. Of these we consider "design the system to protect the user" of primary importance. Our key defense is inherent in the design of RUMINT. We believe multiple semantic windows on network traffic will provide increased insight for the end user as well as an increased obstacle for a potential attacker. As we described in the threat analysis chapter, there are a wide variety of attacks that can occur through a visualization system, but the vast majority target a specific visualization technique. The key defense of RUMINT lies in its diverse range of visualizations. Following closely behind designing the system to protect the user are "assume and intelligent and well informed adversary" and "protect the data generation and data flow." In the design of RUMINT we were well aware that networking protocols

provide little defense to the insertion of malicious traffic by intelligent adversaries. Beyond the diverse visualization defense we just described, we added filtering and encoding capabilities to allow the user to help remove noise and highlight activity of interest. The final design principle "educate the user" is also quite important, but for purposes of the evaluation in the next chapter we chose users with a great deal of network security experience. We believe that both these expert users and those with lesser experience will benefit from future work in the area of "smart books." Smart books consist of visual snapshots of both legitimate and malicious activity in order to enhance understanding and increase performance.

Beyond countering denial of information attacks, the primary design goal of RUMINT is to provide users the ability to view a large number of network packets in a way that supports rapid comparison, deep and broad semantic understanding, and highly efficient analysis. At the same time, we wish to allow intuitive interaction in order to remove noise and highlight packets of interest. It consists of a PVR interface and seven primary visualizations; each designed to provide different semantic windows on network traffic. These include:

thumbnail toolbar (Figure 5.2a): The thumbnail toolbar provides a real-time overview of each visualization window in a thumbnail size display. Doubling as a menu, users may bring up the full size window by clicking on a thumbnail.

scrolling text display (Figure 5.2b): The scrolling text display presents network packets, one per horizontal row, in a user selectable encoding (ASCII, hexadecimal and decimal). It includes a *strings* command, adopted from the Unix environment, that will filter based

on packet contents and display only sequences of characters from the printable ASCII range (e.g. strings of length 3-9).

parallel coordinate plot display (Figure 5.2c): This visualization uses the parallel coordinate plot technique to display scaled values from packet header fields. Currently 19 header fields, up to 19 vertical axes and 19! combinations of headers are supported.

detail display (Figure 5.2d): The detail window displays the selected packet's contents in a traditional hex/ASCII format.

glyph-based animation display (Figure 5.2e): The glyph-based display combines three display panes to animate any two attributes (header fields) of network traffic. The center pane is a two axis parallel coordinate plot and the side panes contain glyphs which move off the screen as the network traffic is processed.

binary rainfall visualization (Figure 5.2f): The binary rainfall visualization displays packet contents, one per line. It has three primary views which map packet contents to display pixels.

scatterplot display (Figure 5.2g): The scatter plot visualization allows users to select any two header fields (19 are implemented) and plots them on a traditional X,Y display. Header field values are scaled to match the dimensions of the display window.

byte frequency display (Figure 5.2i): The byte frequency visualization displays the

presence and frequency of bytes within each packet.

PVR interface (Figure 5.2h): The PVR interface is the heart of the RUMINT application. Packets are captured live from the network or loaded from capture files and stored in an internal cache. The PVR interface allows playback of these packets for viewing in any of the visualization windows. This approach extends the VCR metaphor suggested by Erbacher [32].

From the near infinite space of possible visualization techniques these seven were chosen based on both intuition and feedback from security analysts. During the design process we explored approximately 15 other visualizations, but these were ultimately discarded because they did not most effectively address user needs. We believe these seven visualizations represent a solid set of techniques that can be refined to increase their utility, but it is important to note that the PVR based design of the system scales well. It is a straightforward matter to add new visualizations in a very short time period.

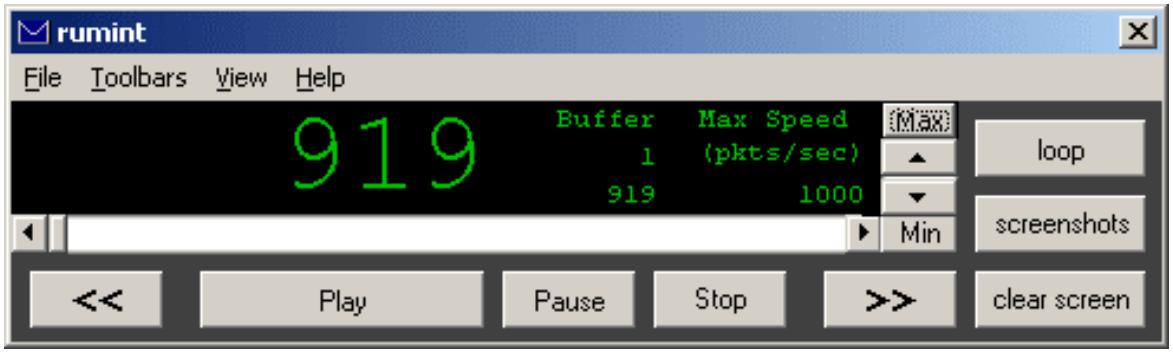


Figure 5.3: The PVR interface is the heart of the RUMINT system. It allows the user to play back traffic from any point in the dataset.

Interaction Paradigm

The PVR (Figure 5.3) is the heart of the RUMINT system and is the primary interface presented to the user. From the view menu on the PVR users may select any of the visualization windows to display playback results. When performing live capture, the captured packets are stored in a RAM buffer and immediately displayed. Playback may be paused to allow closer inspection of previous network traffic and resumed without interruption. The horizontal scrollbar allows the user to select a single packet for display and the << and >> buttons step forward and backward through the traffic. Playback speed is adjustable, using controls on the PVR, from one packet per second to a theoretical limit of 1,000 packets per second. If the underlying system hardware is unable to reach this limit, packets are played back at the fastest possible speed. In addition, the interface includes buttons to continuously loop through the dataset, clear all the visualization windows and to save screen shots. Optionally, users may use the

filtering control panel (Figure 5.4) to filter traffic before it is displayed as well select color encoding. The figure shows two iterations of the filtering and encoding control panel. The initial iteration (Figure 5.4, top) explored the use of custom filters, created by analysts, to interactively remove slices of network traffic from the display. While we believe this is a very valuable approach, we chose to create flexible, general-purpose filters based on packet headers (Figure 5.4, bottom). Ultimately, we believe the best solution may be to allow users to create filters using a general purpose graphical user interface and via the Ethereal's textual filtering language. These filters could then be saved for future use. Further exploration of this topic is a valuable area for future work.

After the user has decided to play back traffic, they can call up the desired visualization window via the view menu on the PVR interface. While this approach is functional at the current level of visualization options, it does not scale well. We designed the thumbnail menu, shown in Figure 5.5 in order to counter this problem. The thumbnail menu contains small-scale display windows that allows the user to see all visualization displays at one time. Whereas a single visualization window may occupy up to the full screen at a given time, the thumbnail menu can display approximately eighty. By observing playback on the menu users can then click on a given thumbnail display to bring up the full scale visualization window. While observing a large number of thumbnail views at one time may present at problem as additional visualizations are added, we believe it is an improvement on the current text-based menu.

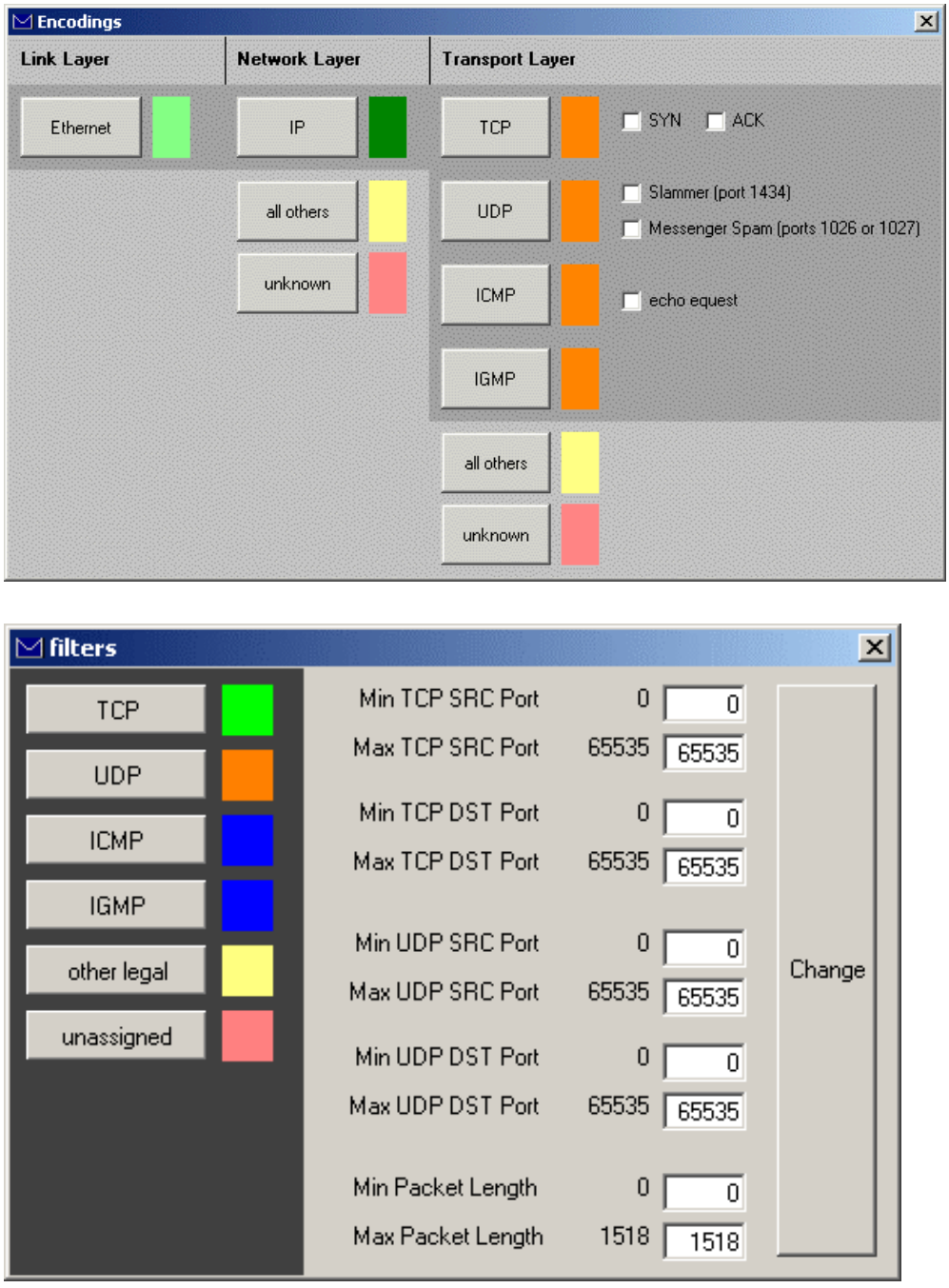


Figure 5.4: Interactive Filtering and Encoding Control Panel. Initial design (top) and revised design (bottom). This panel exploits the fact that filtering and encoding are logically intertwined. It allows the user to perform dynamic queries on the network dataset as well as customize color and other encoding techniques.

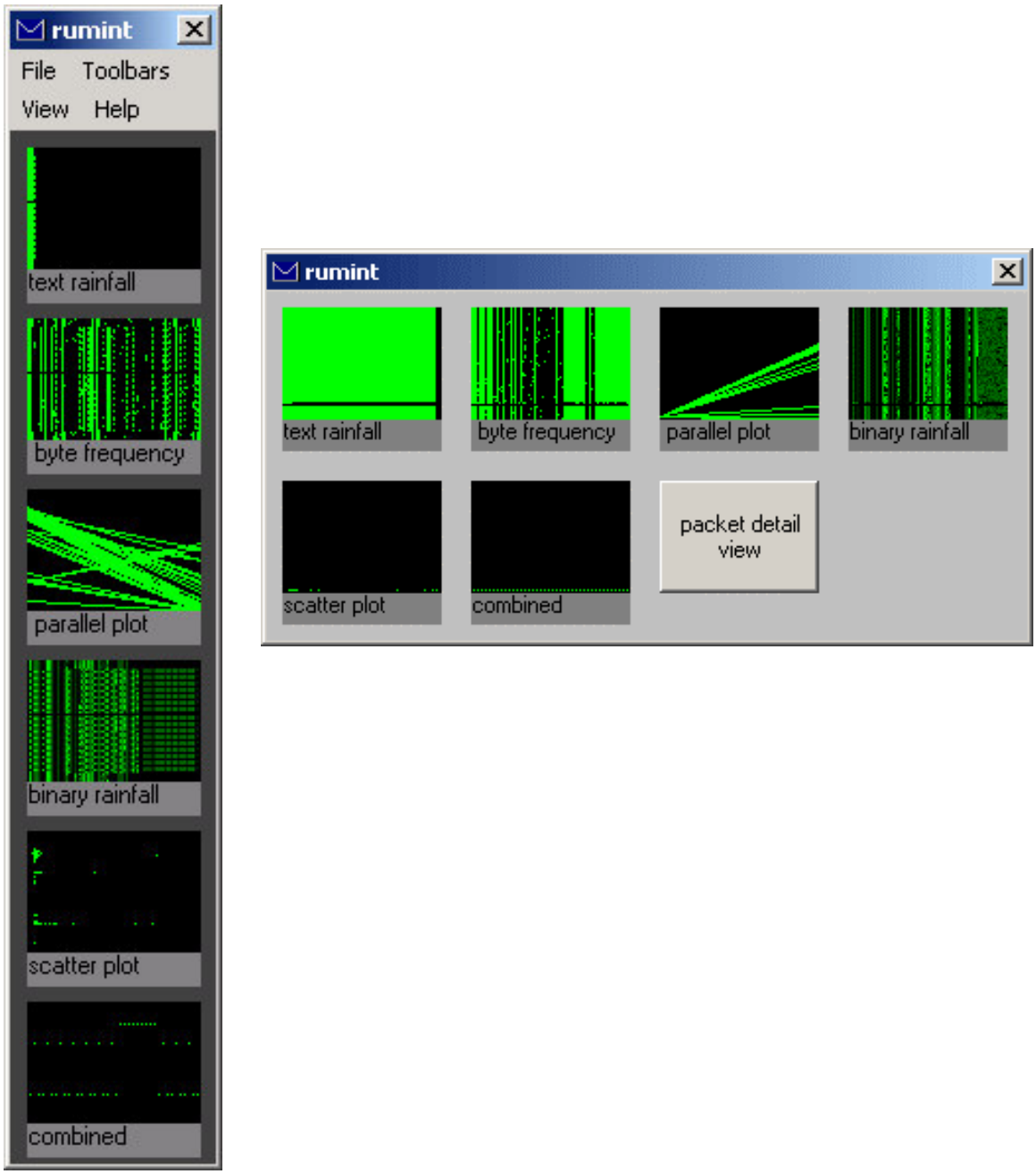


Figure 5.5: Two variants of the thumbnail visualization menu. Initial design (left) and revised design (right).

System Architecture and Implementation

The RUMINT system was developed based upon the framework presented earlier in the chapter. While previous components were implemented using Perl and C, the current iteration was developed using Microsoft Visual Studio, primarily for its strength in rapid GUI development. Our plan was to exploit the strength of GUI development under Windows and port the software to a more robust Open GL / QT application in the Linux/Unix environment after the user interface and visualization design has been finalized. The primary test bed system was an AMD 2500+ with 1GB of RAM, 64MB of Video RAM, 10/100 network card and 160GB hard drive running Windows XP.

RUMINT captures packets live from the network using the winpcap library for kernel level capture and the PacketX [12] ActiveX component for easy access to network data from Visual Studio applications. While this proved straightforward for live capture, PacketX unfortunately does not support pcap file access. This proved problematic and we ultimately resorted to implementing a custom pcap file parser directly in RUMINT.

When the packet data is loaded from a file or captured from the network interface it is parsed and stored in a RAM buffer. The user interacts with the encoding, filtering and zoom menus to select the parameters of interest. The system then queries the RAM buffer for the appropriate packets and submits the data to the graphics engine. The graphics engine updates currently active visualization windows. The following section describes these visualizations.

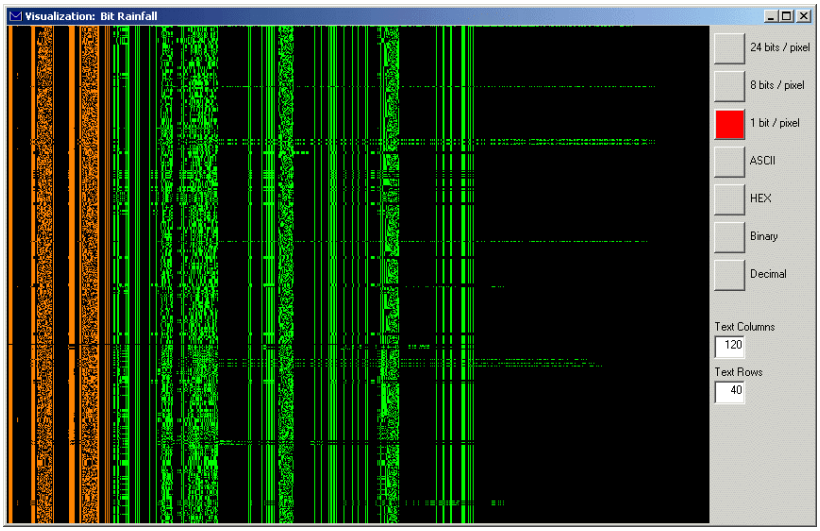


Figure 5.6: Binary Rainfall Visualization of Defcon 11 "Capture the Flag" Network Traffic. One packet is plotted per horizontal line. In this level of zoom, each pixel represents one bit of network traffic. Network layer protocol headers are encoded in orange and the encapsulated IP payload is encoded in green. This visualization allows analysts to readily compare approximately 600-1000+ packets per screen.

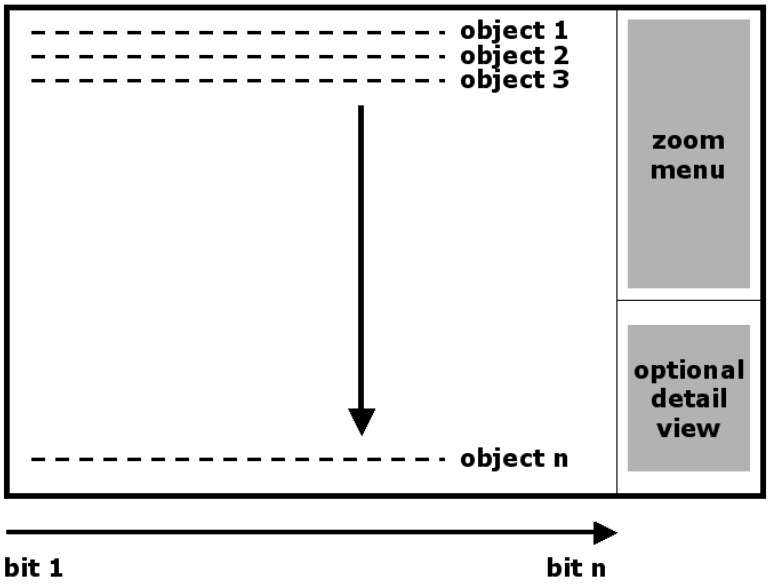
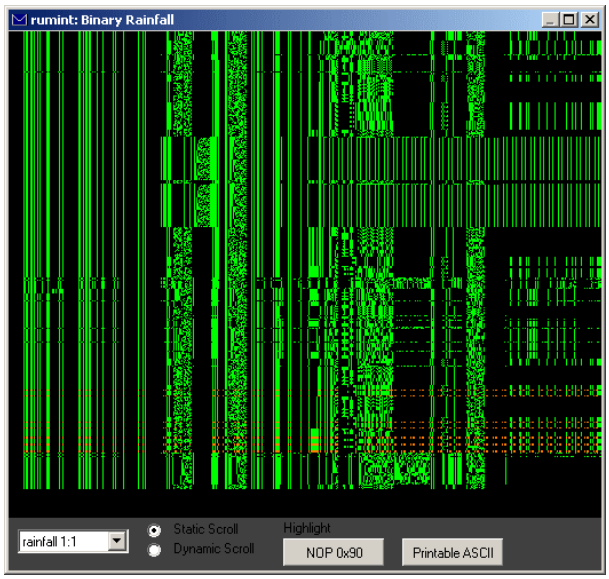
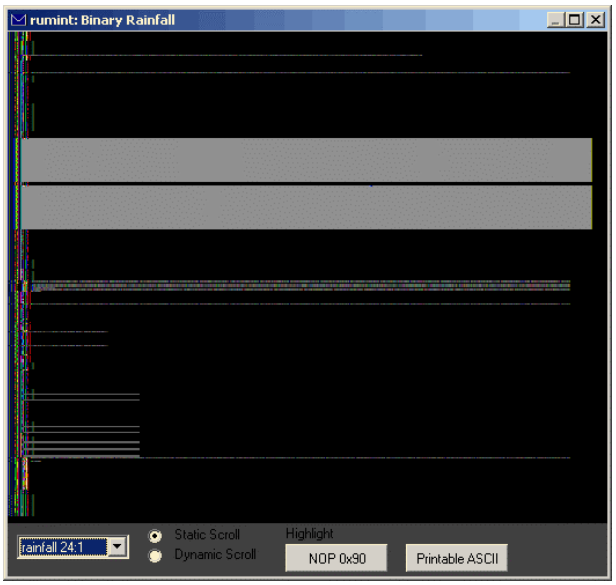


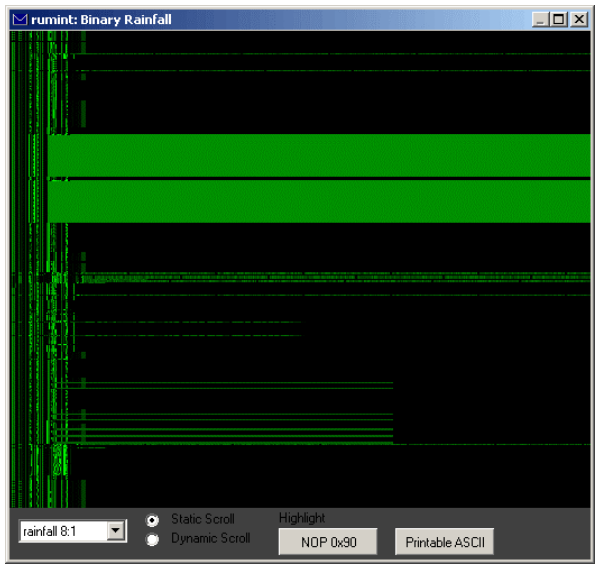
Figure 5.7: Binary Rainfall Visualization Design. The bits of each packet are plotted horizontally. Each new packet is plotted on a horizontal line below the previous packet. The semantic zoom menu is present on the top right. An optional detail view pane, in the lower right, allows viewing of a single packet.



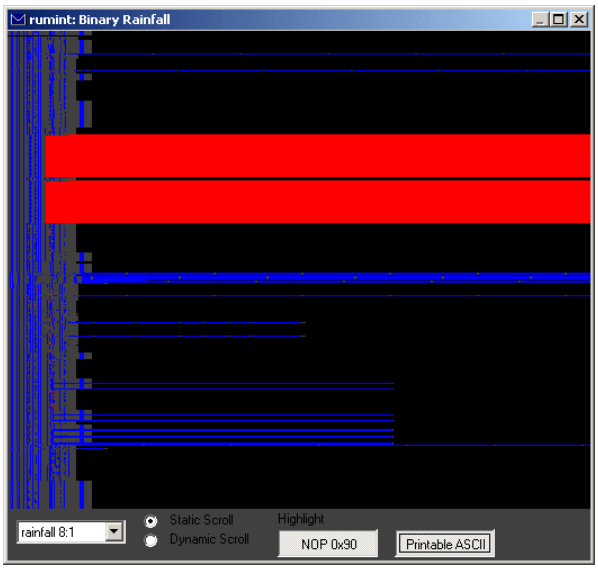
(a) 1 bit per pixel view



(b) 24 bits per pixel view



(c) 1 byte per pixel view



(d) 1 byte per pixel view, with 0x90 bytes encoded in red and printable ASCII values encoded in blue

Figure 5.8: Binary Rainfall Visualization Method. Images (a), (b) and (c) depict three graphical zoom levels. Image (d) uses color to highlight byte values of interest.

Visualization Design

Binary Rainfall Visualization

The binary rainfall visualization (Figures 5.6, 5.7 and 5.8) was inspired by the classic waterfall display used for spectrum analysis but instead plots binary objects, one per horizontal line, in time sequence order. It is useful for a variety of tasks, including comparing the payloads (to include identifying the use of encryption), lengths and headers of packets. When properly color encoded it is useful to locate human readable (i.e. printable ASCII) traffic and shows promise for detecting buffer overflow activity. The user interacts with the display using the semantic zoom menu. We tried placement of this menu in two different ways. Our initial approach was on the right portion of the window (Figure 5.6), but we later moved it to a drop down box as seen in Figure 5.8 and added the ability to change levels by double clicking the display. We believe our more recent approach is more efficient in that it allows more screen space for the visualization, but are aware that it comes at a slight usability cost due to slightly hidden functionality. While conceptually this menu will allow a wide variety of views of the data objects, we implemented seven, four graphical and three textual. The graphical views plot pixels in direct correspondence to the structure of the binary data. These four views include plotting each bit of binary data as a monochrome pixel (Figure 5.8a), each byte of binary data as a 256 level green pixel (Figure 5.8c), each three bytes of binary data as 24-bit RGB pixel (Figure 5.8b) as well as a rainfall representation of byte frequency (Figure 5.9). The fourth graphical visualization, the byte frequency view, plots one packet per horizontal line. Pixels along the horizontal axis, scaled from 0-255, are illuminated based upon the frequency with which the corresponding byte appears relative to each packet.

The pixel may be illuminated as a single color if one or more of a given byte is present (byte presence) or encoded with color based upon the frequency (byte frequency). The textual views allowed the user to view the same objects as ASCII, hexadecimal and decimal representations and will be discussed later in the chapter. We provide each of these views to allow the user to visualize network traffic as needed for their current tasks. As we described in earlier, current techniques for analyzing network payload data rely almost exclusively on textual representations. By combining textual header and payload visualization with graphical techniques, we gain a significant increase in the amount of data that can be displayed on the screen at one time. Table 5.1 shows the increase, in various graphical modes, when compared to ASCII and Hex representations. This comparison is based on the amount screen space required to display an equivalent number of bits of information.

Table 5.1: Comparison of graphical vs. textual information density.

Graphical	ASCII	Hex
1 bit per pixel	15x	45x
8 bits per pixel	120x	360x
16 bits per pixel	240x	720x
24 bits per pixel	360x	1080x
32 bits per pixel	480x	1440x

While the binary rainfall technique is generally applicable to many types of binary objects, we applied the visualization technique to network packets in order to test its effectiveness. A key criterion is that network packets are highly structured and small enough, typically less than 1518 bytes, to be effectively displayed and compared using our graphical modes. We augmented the visualization technique with mouse over information, but believe it would be significantly strengthened with the addition of zoom functionality.

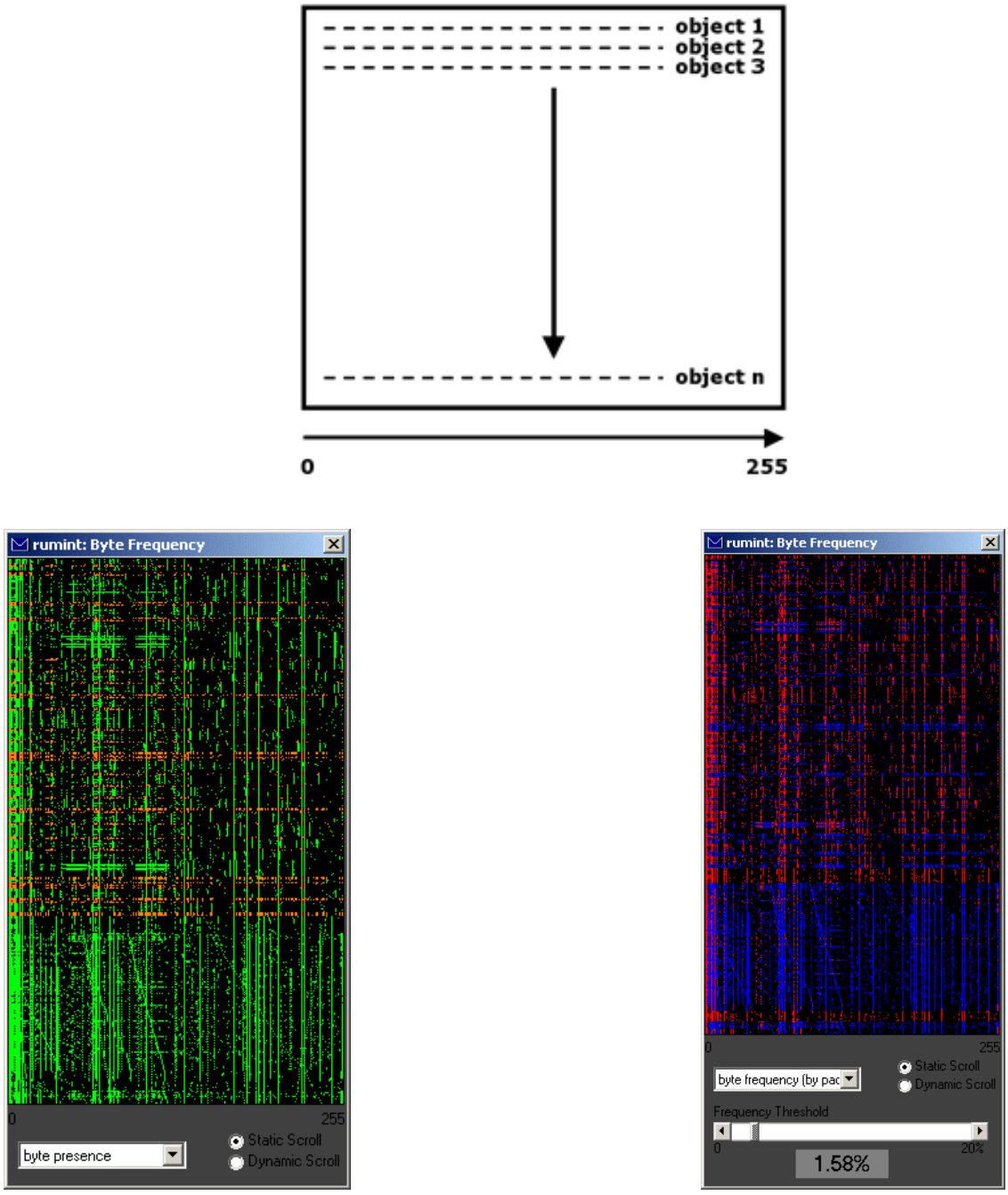


Figure 5.9: Detail of Byte Frequency View: The byte frequency view is one of the semantic levels available to the user. Bytes (0-255) are plotted along the horizontal axis. As each packet is plotted, pixels are illuminated according to the frequency of that byte relative to each packet. The left figure shows the byte presence design and the right figure shows the byte frequency of network traffic from a Honeynet Project Scan of the Month Dataset.

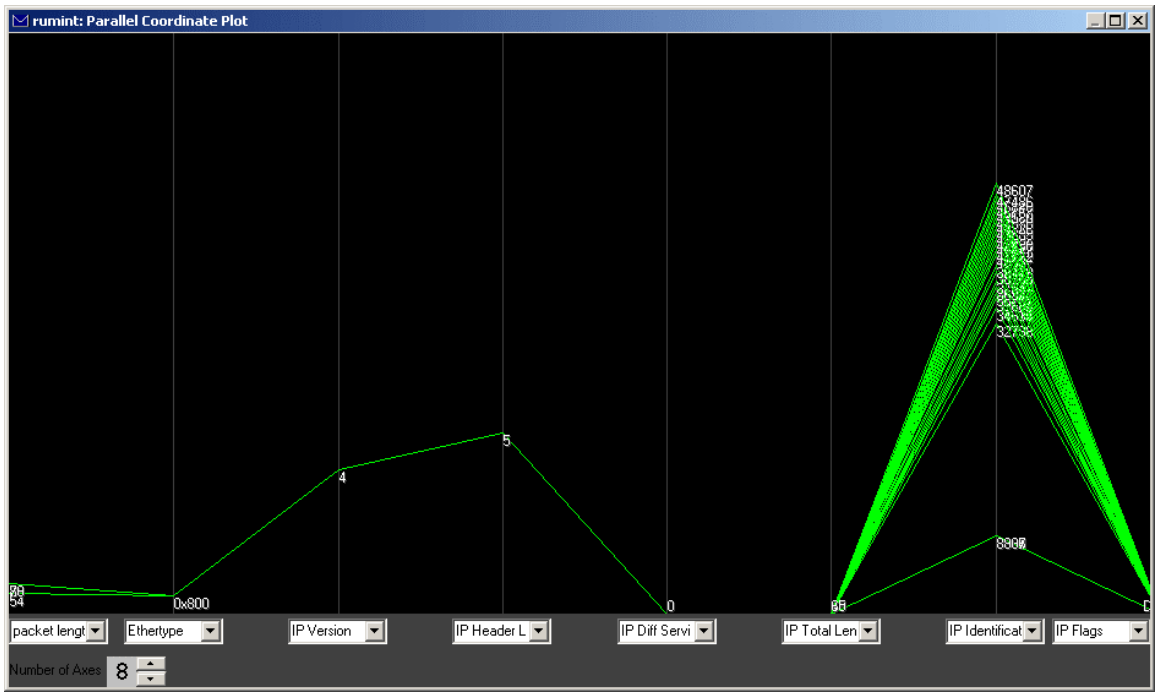


Figure 5.10: RUMINT's Parallel Coordinate Plot Visualization Display. This figure is currently set to display eight of the possible 19 vertical axes.

Parallel Coordinate Plot

The parallel coordinate plot visualization technique is a well-known tool in use by the general information visualization community. We included it in RUMINT because of its ability to effectively display a large number of variables at one time. When coupled with the PVR engine and the other visualization windows it is particularly useful for rapidly characterizing the header fields of packets from large datasets. The current implementation (Figure 5.10) allows the user to interactively set the number of vertical axes as well as independently assign any of 19 header fields to each axis for a total of 19! possible combinations. Because of this capability, analysts can experiment with

alternative combinations of header field to axis mappings. This visualization technique while generally effective, suffers from several shortcomings. Occlusion occurs when line segments and labels are rendered on top of each other. In addition, the graphics library we used (Win32) did not allow us to efficiently implement the ability of a user to click on a given line segment for more detail. We plan to explore solutions to these issues in future work.

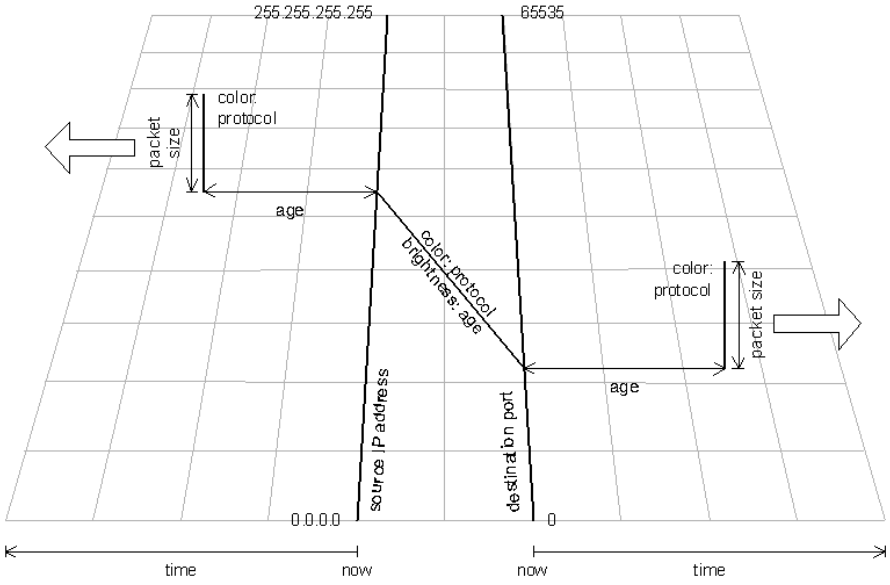


Figure 5.11: Glyph-based Animated Display Design

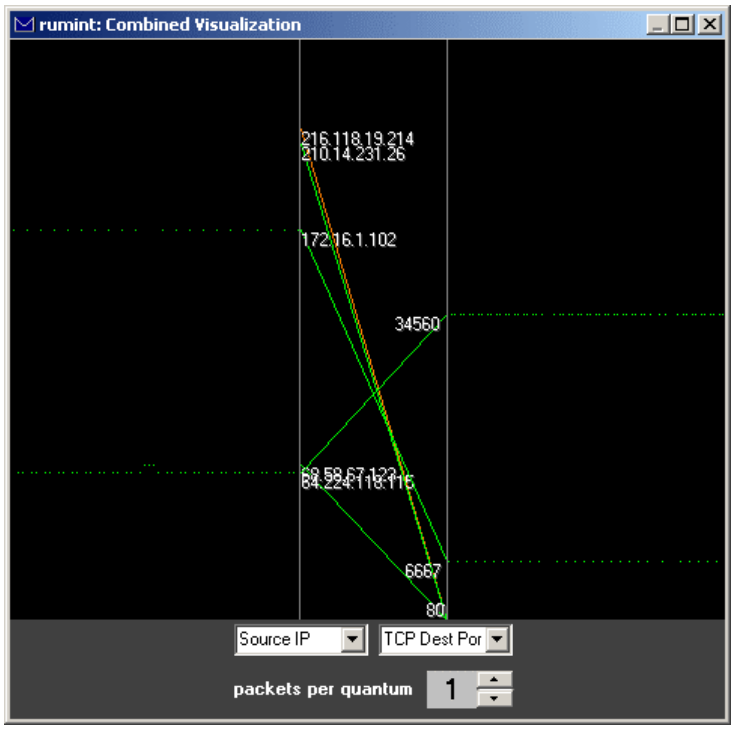


Figure 5.12: Example of Glyph-based Animated Display

Glyph-based Animated Display

The glyph-based animated display (Figures 5.11 and 5.12) combines a two column parallel coordinate plot with glyph-based animation. Similar to our general-purpose parallel coordinate plot visualization, the glyph-based parallel coordinate plot axes can be mapped to any of 19 header fields. The glyph-based approach augments the parallel coordinate plot by creating two animated pixels for each arriving packet. These pixels slide outward across the screen as additional packets arrive. Their vertical position is determined by the end points of the line segment connecting the two parallel coordinate plot axes. We found this technique useful in combination with RUMINT's other parallel coordinate plot visualization to help cope with issues of occlusion. The creation and movement of the glyphs allows the user to observe the precise arrival sequence of hundreds of packets despite occlusion of the parallel coordinate plot line segments. When we prototyped this visualization we created it to support both 2D and 3D using the Open GL graphics library. The primary difference between the 2D and 3D views was the mapping of packet length to the Z axis in the 3D display (Figure 5.11). After prototyping the initial application we integrated it into RUMINT as a 2D display only (Figure 5.12), due to limitations in the Win32 graphics library.

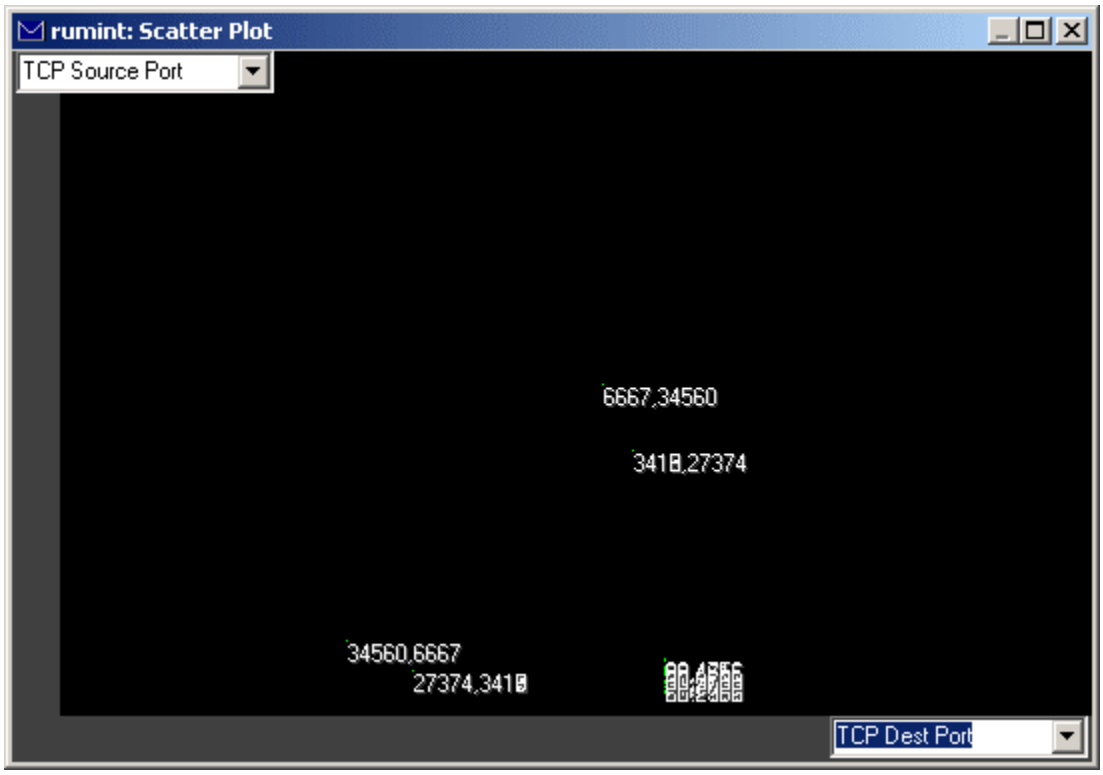


Figure 5.13: Example of Scatter Plot Display.

Scatter Plot Visualization

In order to provide a comprehensive set of tools to the user we included a traditional 2D scatter plot display. Similar to the parallel coordinate plot, it allows users to map any of the 19 header fields to the horizontal and vertical axes. While generally useful it would be strengthened by adding zoom functionality, using techniques to counter labeling occlusion and by techniques that would allow users to select individual packets.

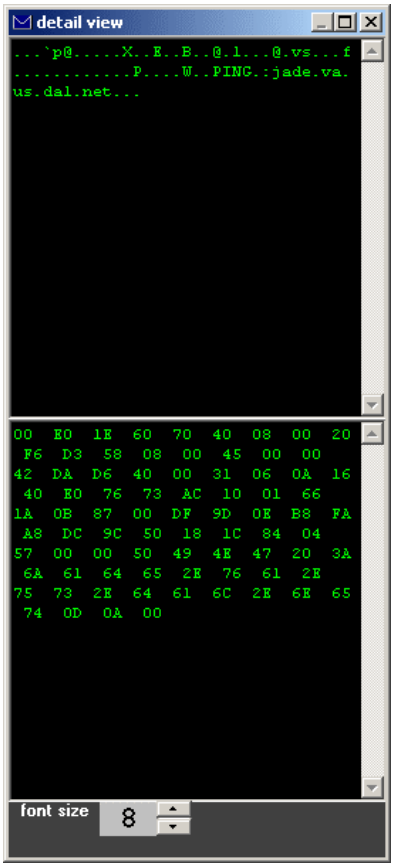
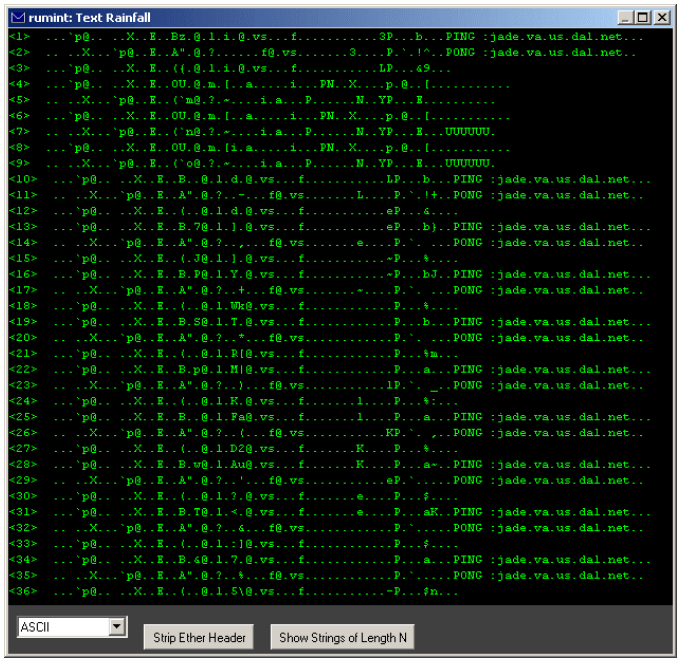


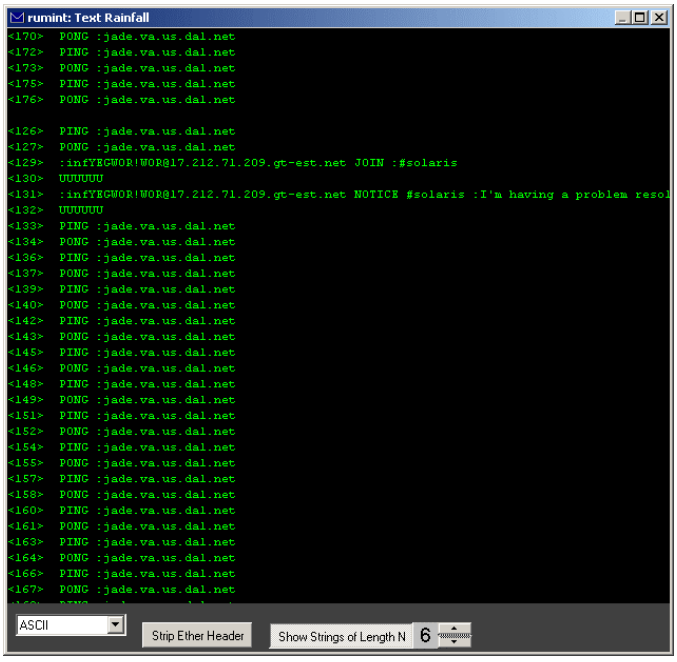
Figure 5.14: RUMINT's Detail View Pane

Packet Detail and Text Rainfall Views

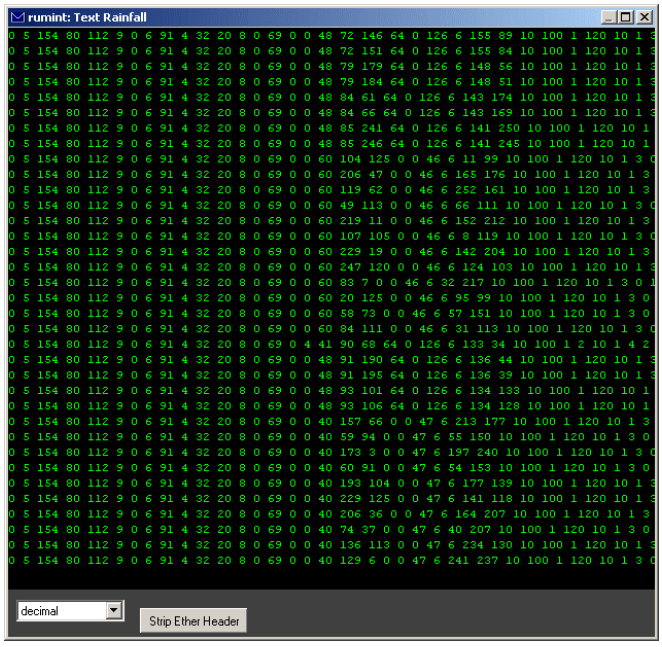
Both of these techniques perform the critical role of providing packet level detail to users. The detail view window (Figure 5.14) uses the canonical hex/ASCII representation commonly accepted as a best practice in packet level analysis. With the addition of the PVR interface's horizontal scroll bar, we have increased its utility by allowing users to rapidly jump to and view any packet in the dataset. A complement to both the packet detail and binary rainfall views, the text rainfall view allows users to observe and directly compare packet contents in hexadecimal, ASCII and decimal representations (Figure 5.15).



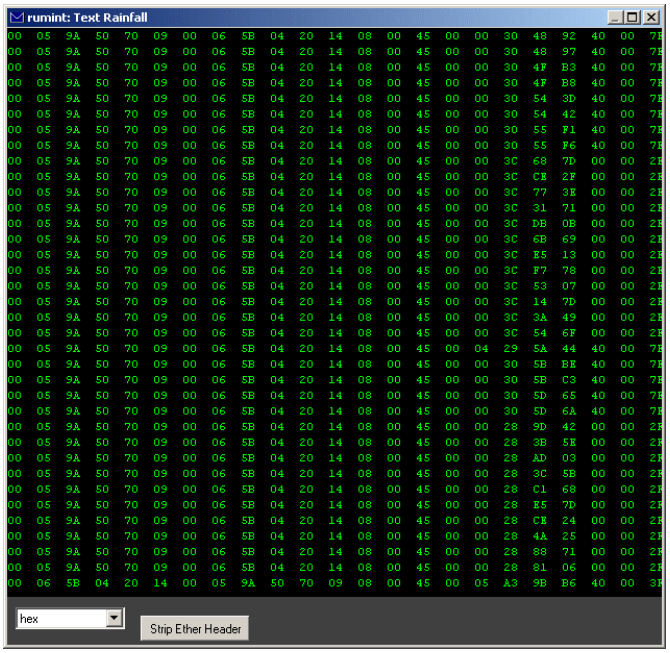
(a) ASCII representation



(b) ASCII representation filtered for printable ASCII strings of length 6 or more

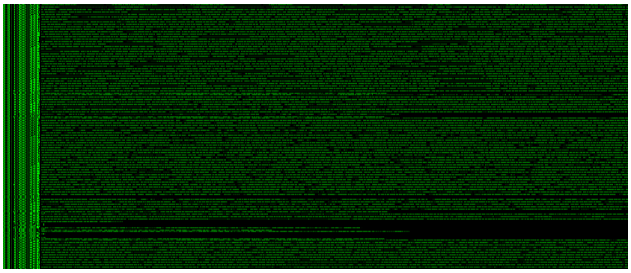


(c) Decimal represtation

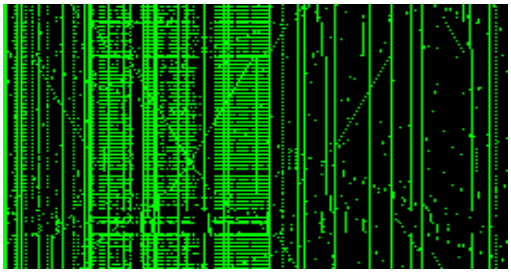


(d) Hexadecimal representation

Figure 5.15: Text Rainfall Visualization. The Text Rainfall Visualization provides three primary representations of the network packet data as well as a Unix-like *strings* command

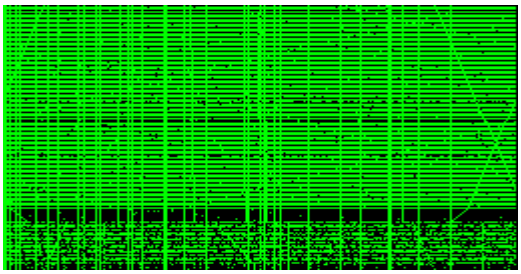


a. ASCII text file retrieved via HTTP



b. ASCII text file byte frequency

c. SSH traffic



d. SSH traffic byte frequency

Figure 5.16: Comparison of ASCII (a & b) and SSH (c & d) network traffic. The binary rainfall images (left column) provide a quick overview of packet structure. Headers and packet lengths are readily apparent. The byte frequency images (right column) clearly show the difference between printable ASCII content (b) and encrypted content (d). Solid vertical lines indicate reoccurring values such as constant header fields. Diagonal lines indicate incrementally changing values, such as sequence numbers.

Example Usage and Additional Analysis

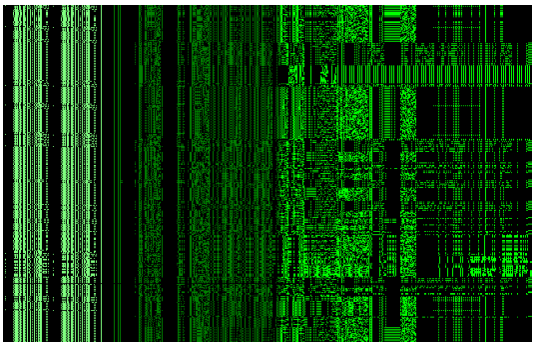
To test the efficacy of RUMINT we used it in a wide variety of scenarios, both operational and experimental, with real-time packet capture and with forensic packet capture files. This section highlights the use of the system in several of these situations.

Characterization of Basic Network Protocols and Coping with Messenger Spam

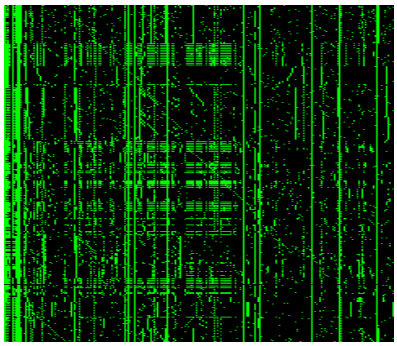
Our initial experiments consisted of testing the system with legitimate SMTP, HTTP, Telnet, SSH, VoIP, FTP and SSL traffic. These tests allowed us to characterize what "normal" traffic looks like (Figure 5.16). We then used the system to visualize a variety of common attack tools in order to create additional visual fingerprints. After conducting this background work, we examined a variety of malicious and non-malicious network traffic from forensic datasets including those from the DEFCON Capture the Flag competition, the United States Military Academy CyberDefense Exercise and the Honeynet Project as well as datasets collected from a Botnet sinkhole created at Georgia Tech.

Operationally, we used the system to monitor Georgia Tech Honeynet traffic for ten months, from July 2004 to April 2005. In a typical usage scenario, users loaded datasets of interest and iteratively adjusted the menu parameters to focus on areas of interest. For example, a user examining a honeynet dataset wished to filter as much Internet background radiation [85] as possible. Being familiar with Pang's observation that a portion of UDP traffic is caused by messenger spam, the user wished to constrain the visualization to display only UDP traffic from common messenger ports then confirm that the traffic was indeed messenger spam and finally to filter those packets. The user first viewed the entire data set and noted that a portion of the traffic contained groups of nearly identical packets (Figure 5.17a) with a high percentage of bytes in the printable ASCII range (Figure 5.17b). The user examined the payloads of these packets and verified their similarity (Figure 5.17c). Using semantic zoom, the user confirmed the

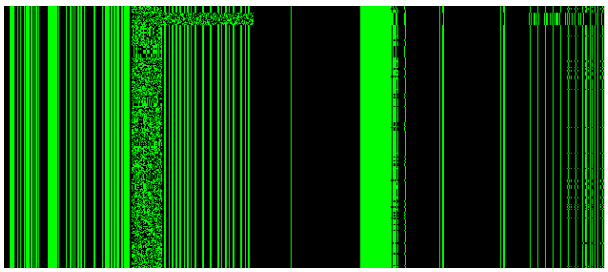
traffic as messenger spam (Figure 5.17d) and created a filter for use with future datasets (the filter can be seen in Figure 5.4, top). We found that our test subjects liked the notion of working through slices of the traffic and then removing them from the dataset. By iteratively removing noise from the display, this approach takes maximum advantage of the high-bandwidth visual recognition capability of human analysts and allows them to incrementally remove known traffic and focus on the unknown, but interesting remainder.



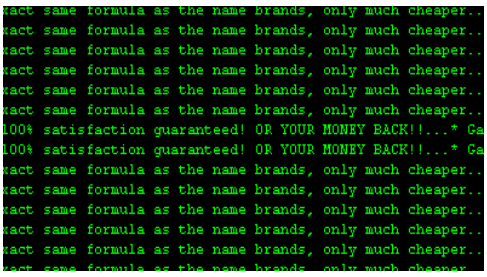
a. View of Georgia Tech honeynet traffic (with headers)



b. Byte frequency of Georgia Tech honeynet traffic



c. Filtering all traffic, but UDP ports 1026 and 1027



d. Closer examination confirms traffic is Messenger Spam

Figure 5.17: Use of the System to Rapidly Identify and Filter Traffic. An analyst views a day's capture from the Georgia Tech Honeynet (a) and examines the byte frequency of the packets (b). Several sets of packets have a large number of bytes in the printable ASCII range. By using the interaction menus, the analyst examines the payloads of these packets and verifies their similarity (c). Using the semantic zoom capability, the analyst confirms the traffic as messenger spam (d) and creates a filter that can then be used with future datasets (the filter can be seen in Figure 5.4).

This scenario relied heavily on the Binary Rainfall visualization. We were pleased with its overall design. Its primary strength is the lack of abstraction, making it ideal for low-level analysis of network packets and other binary objects. The byte frequency display augments the bit and byte level graphical views and provides insight into the nature of the traffic. We found it most useful to rapidly classify traffic as human readable, machine readable or encrypted. Both techniques effectively allowed over 1,000 network packets to be viewed at a very detailed level and easily compared. The exact number of objects that can be displayed approaches the vertical resolution of the user's monitor. This upper limit proved to be sufficient when combined with zooming and filtering, but we envision the need to allow the user to page, or scroll, through multiple screens of content rather than the confine them to the single view window that our current system provides. While the system succeeded in visualizing a large number of objects, the rainfall visualization is limited to displaying approximately 1000 bits (1 bit / pixel mode), 1000 bytes (8 bits / pixel mode) and 3000 bytes (24 bits / pixel mode) of each packet at any given time. In some instances this might prove to be an obstacle, but in the case of network packets it was successful, particularly when compared with a pure ASCII/Hex text display. Ethernet frames are limited to 1518 bytes which is well within the display parameters of the system. In future implementations we will also consider the addition of a traditional, i.e. non-semantic, zoom capability to allow far more of the data to be displayed. It is important to note that test users found the 1 bit and 8 bits per pixel modes most effective. The 24 bit mode proved useful for comparing packet length, but little else. In future work we plan to test combined modes that use a higher resolution (1 bit and 8 bits per pixel mode) display for header data and lower resolution (8 and 24 bits per pixel) display for payload data.

While the visualization provides high information density it comes at the cost of lost timing data. Packets are visualized as they arrive, which results in an information rich display that is useful for comparing packets. Spacing packets based on time of arrival is easily possible, but would create a sparsely populated display that we believe would hinder analysis.

It is also important to consider the security of graphical binary rainfall screenshots. Because the graphical modes represent large amounts of actual network traffic, to include payloads, it would be straightforward for an attacker to extract this information by creating an application which examines pixels. The same amount of care should be taken with these images as when sharing network capture files. In future work we will consider creating anonymization tools for the images similar to tcpdpriv and ipsumdump. We also believe that many classes of information visualization tools, including our system, are subject to overt and covert manipulation by malicious entities who inject carefully crafted traffic into the network (see Chapter 4).

Effective interaction is key to the value of the system. While the binary rainfall visualization technique allows users to compare 600-1000+ binary objects and detect general patterns, anomalies and outliers, it is most effective when combined with dynamic queries, semantic zoom and interactive encoding. Our users used these capabilities to focus on areas on interest, but an unpredicted filtering issue emerged. Our system design failed to take into account the user's need to toggle between a filtered range and its inverse. Users desired this capability to examine certain classes of traffic and then remove that slice of data from view. Essentially, what was required was the ability to invert each filter. While we were able to hard code this capability in select instances, we envision, in future systems, the utility of a filter database where analysts

can build and share these filters (perhaps with analyst comments). Each filter would have three states: off, on (band pass) and on (inverted band pass).

System performance was acceptable up to 100,000 packets, but was sluggish beyond this level. There were two primary reasons: slow graphics operations and inefficient data structures to buffer packet data. We were not surprised by these limitations. The strength of Visual Studio lies not in efficient graphics operations and pointer-based data structures, but in rapid GUI development. In the future, we believe we can achieve an order of magnitude performance boost by utilizing Open GL and more efficient data structures.

Beyond the characterization of typical network protocols, we examined the fingerprints left by common network attack tools. For this work, our experiments were conducted in a networking laboratory and we gathered data using the following scenarios: baseline "normal" traffic, attacks using single tools without extraneous traffic and attacks using single tools with typical traffic. Our intent was to test how well our candidate visualization techniques performed with and without the noise of routine traffic. We plan further experiments that test less aggressive tools and multiple tools in parallel with and without routine traffic.

The system and visualization suite was tested with a range of popular network attack tools falling into two broad categories: network reconnaissance and vulnerability assessment. The network reconnaissance class of tools typically allow ping sweeps, TCP/UDP port scans and operating systems detection. Many high quality tools of this class are freely available and widely used by attackers. Network assessment tools probe target machines for unknown vulnerabilities. To test the efficacy of our approach we utilized the tools and host operating systems listed in Table 5.2.

Table 5.2: Attack Tools Tested

Tool	Attacker OS
nmap 3.0	Windows XP, Redhat 8
nmap 3.5	Windows XP
nmapwin 1.3.1	Windows XP
Superscan 3.0	Windows XP
Superscan 4.0	Windows XP
scanline 1.01	Windows XP
nessus 2.0.10	Redhat 8
nikto 1.32	Windows XP
sara 5.0.3	Redhat 8

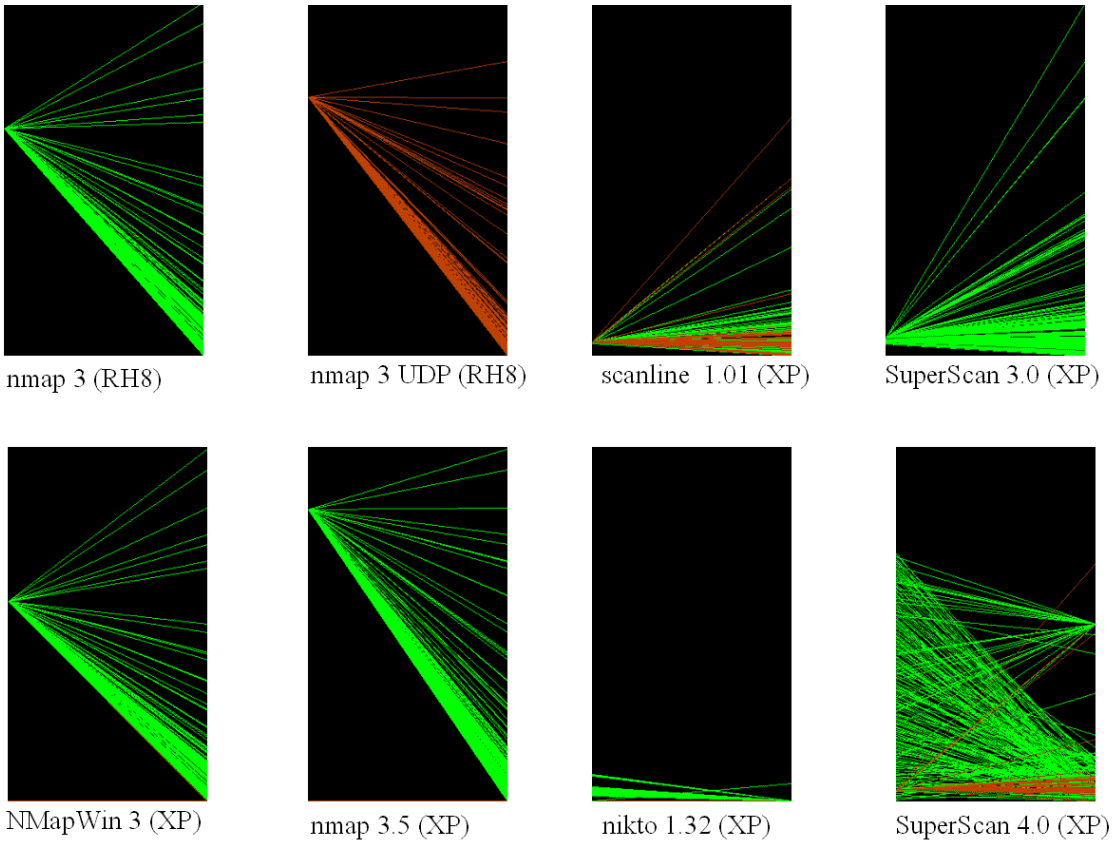
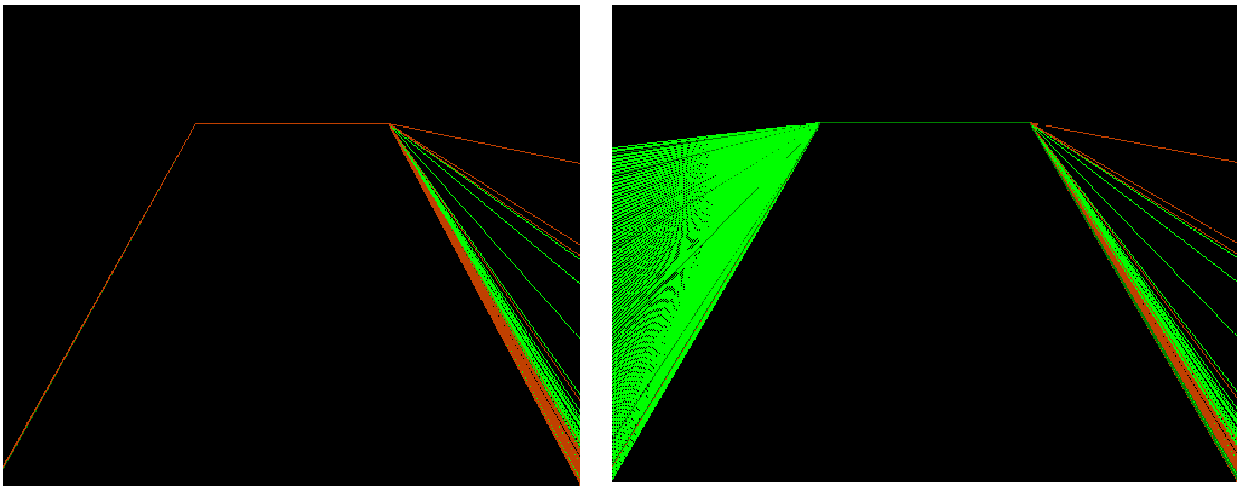


Figure 5.18: Attack Tool Fingerprints (External Port to Internal Port)

To varying degrees, all the visualizations we tested proved effective in analyzing and fingerprinting attack tools. In particular, the port-to-port parallel plot view proved to be of significant value. The images in Figure 5.18 dramatically show the differences and similarities between several tools run from both Linux and Windows XP operating systems. UDP traffic is in orange and TCP traffic is in green. Each fingerprint can be reliably reproduced with each subsequent use of the tool with only slight variations in the location of the attacker's source ports. The default target ports remain the same. Some

may argue that these tools are flexible and alternate ports may be chosen by the attacker. In addition, some tools have publicly available source code and an attacker could create a heavily modified application and thus alter the fingerprint. This is true, but naive use of default settings would indicate that the attacker might be of limited experience. In addition, some tools do not have publicly available source code and are not easily modifiable. Another insight is that by knowing the attack tool in use, the network administrator can take appropriate action. For example, if your web server was probed using the nikto vulnerability assessment tool (figure 5.18) the system administrator might wish to do the same in order to be certain that the tool did not report any vulnerabilities.



scanline 1.01 **Superscan 4.0**

Figure 5.19: Comparison of the network behavior of scanline 1.01 and Superscan 4.0. Similar network behavior may indicate code lineage. Both pictures are of a parallel coordinate plot depicting the external port, external IP, internal IP and internal port axes. UDP traffic is encoded in orange and TCP traffic is in green.

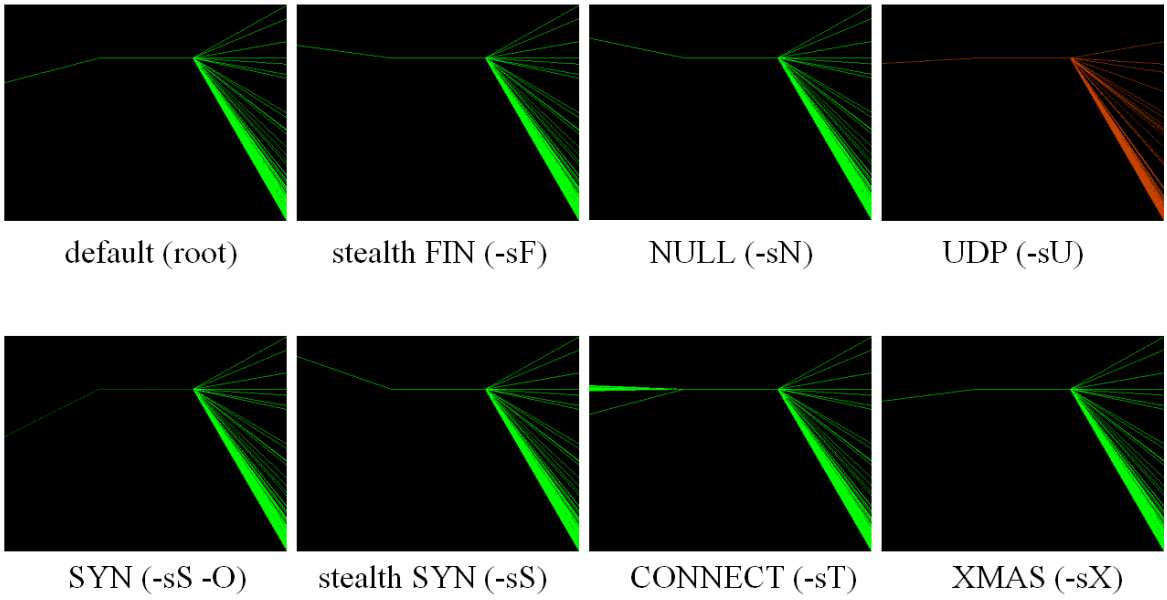


Figure 5.20: In-Depth Look at Common nmap Options

The external port/IP to internal IP/port coordinate plot views in Figure 5.20 compare common modes of nmap 3.0. The respective mode and command line switch is listed underneath each image. After studying these images we noted several things. This view is useful for normalizing the characteristic nmap fan because all attacks against the same IP address show the base of the fan at the same point. Stealthy attacks that take advantage of weaknesses in the TCP protocol, such as the SYN scan, still need to send packets across the network and the signature is still visible. While it is still possible to take advantage of different implementations of the TCP/IP stack to perform evasion or insertion attacks, the visualizations above show that some classes of stealth techniques can be detected. Aspects of the underlying implementation and operating system show

through as well. If you consider the range of source ports used by each mode you will see that there is a difference. Nmap typically relies on raw sockets allowing the application to control virtually every aspect of packet construction. The CONNECT scan shows a wide range of source ports in use that we suspect is due to reliance upon the connect system call. The ability to predict operating system source ports was recently proven to be a critical component of TCP reset attacks. A weakness of the visualization is the inability to detect subtle differences between most of the scans. The FIN, NULL, XMAS, SYN and SYN with operating system fingerprinting all appear the same. In future work we plan to develop visualizations that show the flags in use by TCP packets as we believe that this will show an attacker's operating system fingerprinting attempts. Reliance upon the same code base may also be evident in some cases. This evidence might prove useful with such tasks as quickly estimating if two malicious software applications were created by the same person or same malware toolkit. Figure 5.19 (left) shows the scanline 1.01 tool and Figure 5.19 (right) shows Superscan 4.0. Both tools were provided by the same company with scanline being made available some time before Superscan. The port scanning fans are virtually identical, but the source port fan is dramatically different. We suspect that Superscan was developed from the same base source code, but with the addition of multithreading. We are unable to confirm this, as source code for these tools is not available.

Examining Campus Backbone and Honeynet Traffic Using the Glyph-based Animated Display

Backbone Traffic

Visualization of enterprise level networks is an active area of research. While we have been focusing primarily on packet level analysis, we tested the glyph-based animated display with traffic captured on the Georgia Tech campus network. This network supports 2.5 class B address spaces, approximately 30,000 active computers and has an average throughput of 600 Mbps to the Internet. Figures 5.21, 5.22 and 5.23 show results from our experimentation: inbound campus backbone traffic (5 second snapshot), outbound campus backbone traffic (5 second snapshot) and inbound campus network traffic (10 millisecond snapshot). In analyzing the images, we believe we have generally exceeded the capability of the animated technique. Creating and animating two glyphs for every network packet is extremely resource intensive and exceeds the capability of desktop workstations at this level of network activity. In addition, scaling the network address spaces and ports down to screen resolution caused a great deal of occlusion. Despite these shortcomings, our experimentation did produce some insights. By examining the 5 second captures in Figure 5.21 vertical striations are evident. We believe these were caused by the inability of our commercial network sensor to capture packets at this speed. Also evident from the images and from our observation of the animated playback is the general relationship between the ports in use. For example, Figure 5.21 shows a great deal of inbound traffic arriving at low port numbers, indicative of Microsoft Windows machines.

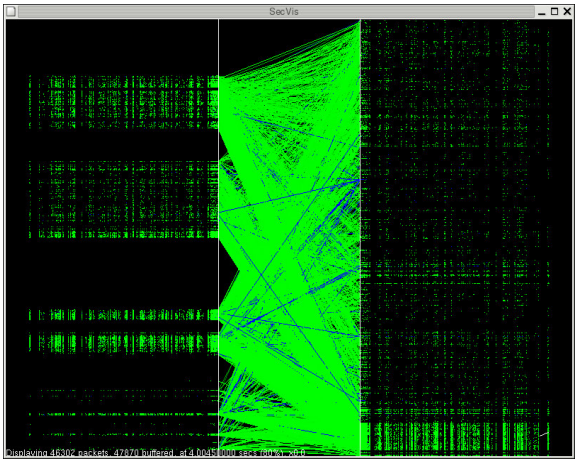


Figure 5.21: Inbound Campus Backbone Traffic (5 sec) The left vertical axis depicts IP address on the Internet and the right vertical axis depicts the destination port on Georgia Tech machines.

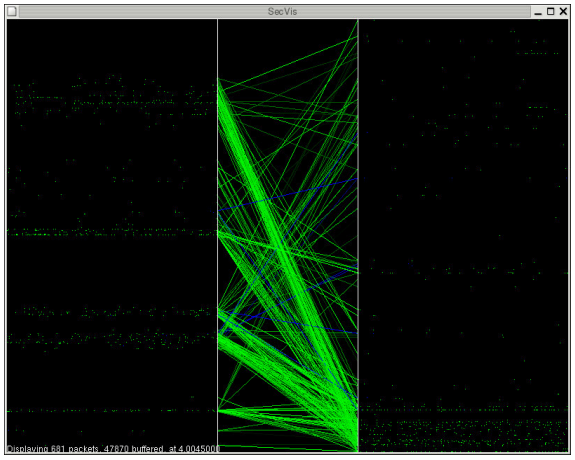


Figure 5.22: Inbound Campus Backbone Traffic (10 msec). The left vertical axis depicts IP addresses on the Internet and the right vertical axis depicts the destination port on Georgia Tech machines.

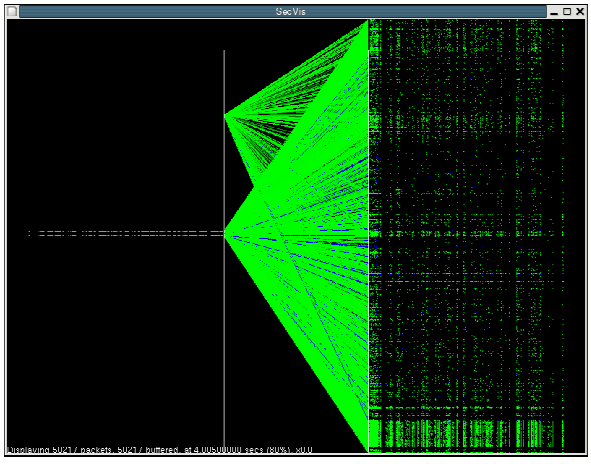


Figure 5.23: Outbound Campus Backbone Traffic (5 sec). The left vertical axis depicts source IP and the right vertical axis depicts destination port on external Internet servers.

Honeynet Traffic

A honeynet is a network of honeypots, machines that are intended to be compromised, designed to provide the system administrator with intelligence about vulnerabilities and compromises within the network. A honeynet is placed behind a reverse firewall that captures all inbound and outbound data. The reverse firewall limits the amount of malicious traffic that can leave the honeynet. This data is surreptitiously contained, captured, and controlled. Any type of network device can be placed within the honeynet, to include configurations identical to production machines elsewhere on the network. These standard production systems are used on the honeynet in order to give an attacker the enticing look and feel of a real system. Since honeypots do not offer any legitimate services to Internet users and the Internet addresses of the honeypots are not publicly known, most traffic on the honeynet is suspicious. These qualities result in an unusually high signal to noise ratio. We exploited this capability to test the effectiveness of our approach by applying the glyph-based visualization to a variety of archival honeynet capture files. As an example, on January 25, 2003 the Slammer worm hit the Internet and our campus network. Slammer targeted Microsoft SQL Servers and machines running the Microsoft SQL Server Desktop Engine. The worm sent out its exploit code in 404-byte UDP packets to port 1434. Figure 5.24 shows 640 seconds of this traffic. Clearly visible are many identically sized UDP packets (blue) targeting a single port. As another example, on February 22, 2004, a computer on campus compromised one of the honeypots. Honeynet traffic jumped from 1.5 MB on February 21 to 4.5 MB on February 22 and back down to 0.5 MB on February 23. It took several days of manually studying the logs to identify exactly when the compromise occurred. Figure 5.25 demonstrates that

using the visualization system, the compromise traffic is easily observed; the timestamp in the display directs the analyst to the exact point in the capture log to begin detailed analysis. The left IP plane shows that all traffic occurs inside the Georgia Tech address range. This significantly reduces the time required to conduct forensic analysis. This reduction in effort is a key benefit of using our visualization system. Security analysts are faced with a tremendous amount of security logs and our system allows them to rapidly identify regions of interest. A network administrator using this tool could also identify this type of abnormal traffic in real time using our visualization system.

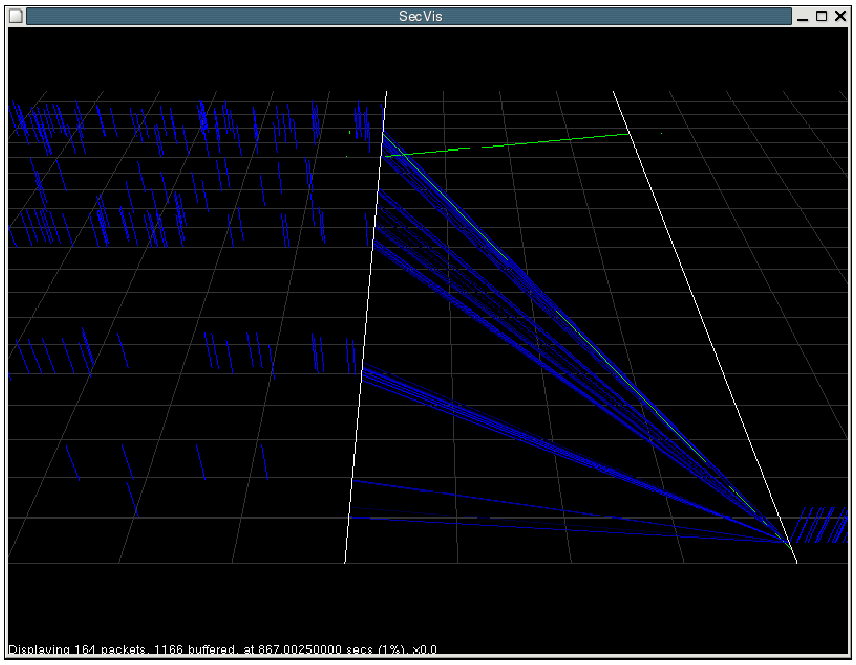


Figure 5.24: Honeynet Traffic during the Slammer Worm Attack

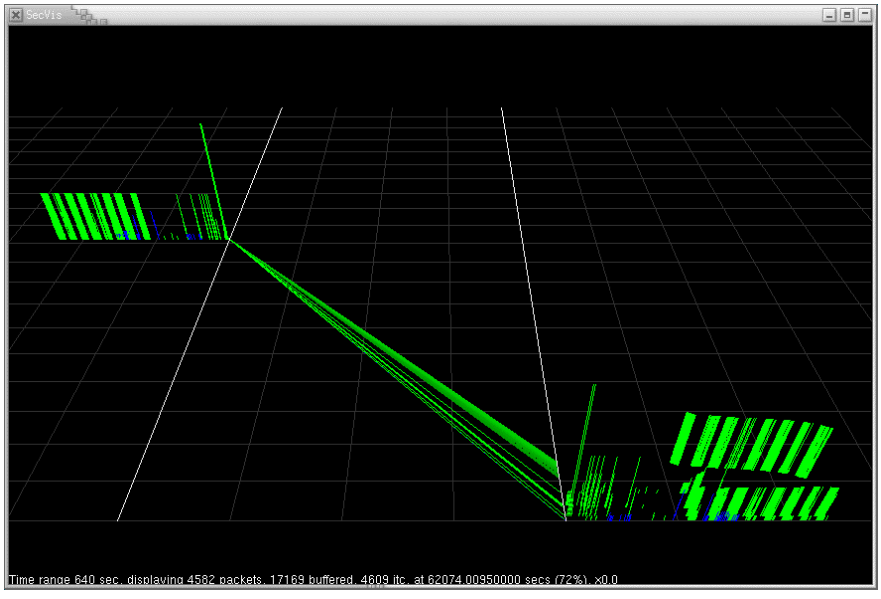


Figure 5.25: Honeynet Compromise Traffic

CHAPTER 6

HUMAN-CENTRIC EVALUATION OF RUMINT

Security visualization is a relatively new area and while initial results bear great promise, there has been a noticeable lack of human-centric evaluation to determine its true effectiveness. We argue that human-centric evaluation is of critical importance in determining the effectiveness of visualization solutions to denial of information problems. By definition, security visualization systems rely upon the perceptual system of humans. They are designed to help solve human tasks. Without including a human-centric evaluation, based on user requirements, a key component is missing. Unfortunately, evaluating humans is messy, time consuming and difficult. Humans do not provide clean results and it takes large numbers of participants to produce statistical significance. Where in traditional system analysis thousands of iterations of an algorithm may be run over a weekend to rapidly converge on meaningful results, we do not have that option. We have to convince test subjects to voluntarily participate. Each hour they spend is an hour they lose for other activities they wish to pursue. In short, human evaluation does not scale. At most research facilities we must comply with strict human subject study requirements overseen by institutional review boards. All organizers must undergo formal certification in human subject experimentation before they can begin work on a specific project. Initial approval typically takes weeks or months. Each change in the study protocol requires re-approval. Instead of a simple change to a program and immediately rerunning of tests, our evaluation cycle takes several orders of magnitude more time. System design and evaluation require two, typically distinct, skill

sets: a deep understanding of the security problem and a thorough knowledge of human evaluation techniques. Despite these challenges, we must seek acceptable evaluation of our security visualization systems.

To be most accurate, security visualization evaluation requires two key components: real-world user requirements and an evaluation using subjects as close as possible to real-world users. The linkage between these two components then allows for an evaluation to determine how well these requirements have been met. The goal of this paper is to explore these two facets by presenting results and lessons learned from the requirements generation phase of a security visualization system design and from two subsequent quantitative evaluations. We address the problem of security analyst information overload and evaluate the performance of the RUMINT packet visualization system [90] in meeting user requirements. Our user attitude and needs assessment survey explored the limitations of two widely employed network analysis tools: the Snort intrusion detection system and the Ethereal packet level protocol analyzer. Specifically we sought expert level insight into the following questions:

- What is the difficulty of inserting malicious data into security related datasets in three contexts: intrusion detection alerts, network packets and system logs?
- Given some degree of malicious data, how difficult is it to mislead or confuse the human when using Snort and Ethereal?
- What is the threshold at which Snort and Ethereal become difficult to use? Specifically, we sought to quantify the threshold at which these tools overwhelm the human operator. We did not address traditional hardware

issues such as graphical frame rate and CPU utilization, only the cognitive burden placed on the human.

- What are the general strengths and weaknesses of these two tools?

From these requirements, we designed two quantitative experiments to evaluate human performance using RUMINT. We focused specifically on packet level analysis and did not address intrusion detection alerts and system logs. We sought to answer the following questions:

- Can humans visually detect malicious activity in the playback of forensic network traffic?
- How precisely were analysts able to locate malicious anomalies within the dataset?
- How long does it take an analyst to detect and precisely locate malicious activity?
- What classes of malicious activity were most easily detectable?
- How much task specific training and analyst experience is required to accomplish these tasks?
- During the course of using the tool, how rapidly does human performance improve?

Instead of being weighted toward students, our requirements study was weighted heavily toward professional security analysts who were recruited during the Black Hat USA 2005 and DEFCON 13 security conferences. In the second phase, our two

experiments were hands on and included 10 Master of Science in Information Security Students and 22 upper level Bachelor of Science in Electrical Engineering students. Both experiments were conducted at a large research university. During the course of our analysis we provide lessons learned to help support other researchers as they seek out operator requirements and design human-centric evaluations of their systems. We do not discuss the design and implementation itself, as these issues have already been addressed in Chapter 5.

The primary contributions of this chapter include our survey of expert user attitudes regarding the strengths and weaknesses of two common analysis tools Ethereal and Snort, the vulnerability of intrusion detection systems, the Unix-style system logging service (syslog) and network packet captures to the insertion of malicious data and an examination of the human-centric threshold at which analysts become overwhelmed when using these tools. Our contributions also include results, analysis and lessons learned from two quantitative experiments that test the ability of intermediate and advanced users to both detect and precisely locate security anomalies in network traffic.

User Requirements and Attitude Survey

Our user requirements survey was designed to elicit feedback from experienced security practitioners in order to guide the design of our prototype visualization system. We primarily recruited participants at the Black Hat USA 2005 and DEFCON 13 computer security conferences. We chose these venues due to the high percentage of experienced computer security analysts from industry. Ultimately 39 individuals participated with 25

from industry/government and 14 from academia. Members of the industry group were 64% professional security analysts, the remainder were in related fields (system administration, network security assessment, software testing) with significant security responsibilities. All participants from academia specialized in computer security.

We asked survey participants to rate their background and knowledge level in computer security. We were pleased with the results. Most reported a significant background in computer security. On a scale from 1 (other than security) to 6 (exclusively security), the average was 4.846, with a standard deviation of 1.136. Likewise, the group reported a computer security knowledge level of 4.743, with a standard deviation of 1.093, on a scale of 1 (novice) to 6 (expert). Based on our interactions with the participants and their self-reported skill assessment we believe we met our goal of surveying experienced security analysts.

Vulnerability to Malicious Data

The core of the survey included three main sections. The first section sought to determine security analyst attitudes regarding the likelihood that security visualization systems would be targets of attack as well as to assess the vulnerability of several key security data sources upon which these systems depend: system logging, intrusion detection and packet capture.

Table 6.1: Ability of Attackers to Inject Malicious Data into pcap, syslog and Snort Datasets

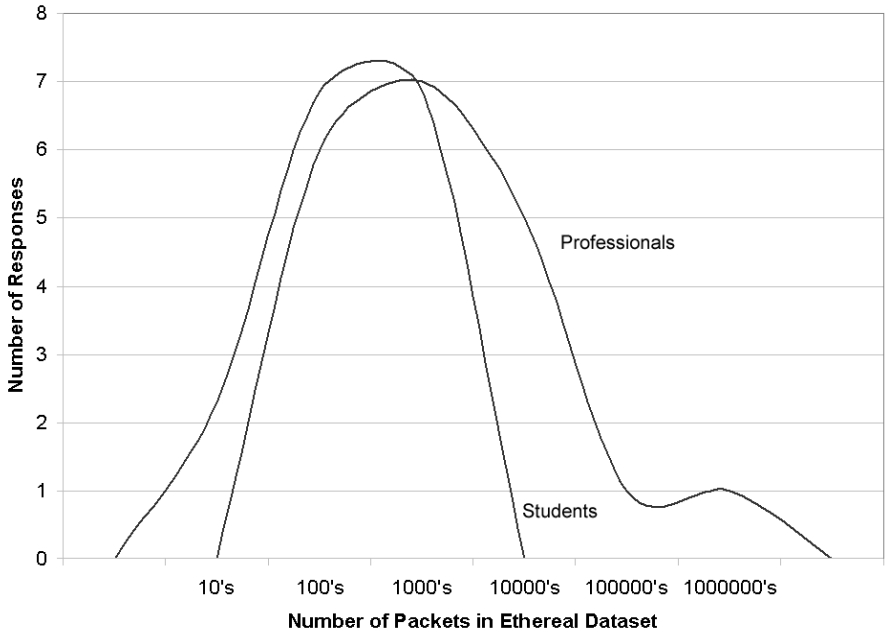
Question	Scale	Average	Standard Deviation
Attackers can insert malicious data into packet captures	1 = No 6 = Absolutely	5.263	1.032
Attackers can insert malicious data into system logs remotely (across a network)	1 = No 6 = Absolutely	4.842	1.462
Attackers can insert malicious data into system logs from a local user account	1 = No 6 = Absolutely	5.184	1.136
Attackers can remotely (across a network) trigger desired intrusion detection system alerts	1 = No 6 = Absolutely	5.436	1.071
Information visualization systems can best be used for critical or non-critical applications	1 = Non-Critical Applications 6 = Critical Applications	4.789	1.044
Information visualization systems are likely targets for attack	1 = No 6 = Absolutely	4.237	1.618

As shown in Table 6.1, respondents felt overwhelmingly that attackers can insert malicious data into three data sources including from remote locations across a network. We concur with their assessment. The ability to insert malicious data is a vulnerability that enables attackers to target both automated security processing systems, such as intrusion detection systems, as well as the humans utilizing these systems. We are concerned particularly with the human-centric aspect because it is directly applicable to DoI attacks that seek to mislead or overwhelm human analysts. An excellent example is the use of malicious network packets to trigger arbitrary alerts from the Snort intrusion detection system. Just a small number of carefully chosen alerts can consume the limited resources of individuals charged with monitoring intrusion detection systems and defending the network. Our security visualization system, which we evaluate later in this chapter, is designed to help counter packet-level attacks by providing analysts new tools to interpret and analyze data, effectively increasing the difficulty for a malicious entity seeking to achieve a successful Denial of Information attack. We are less certain of the validity of the final two questions listed in Table 6.1. We sought to determine if security experts believed information visualization systems would be considered a high priority target for attack. While the survey results indicated the respondents believed they would be used for critical applications and likely targets for attack, informal discussions in the

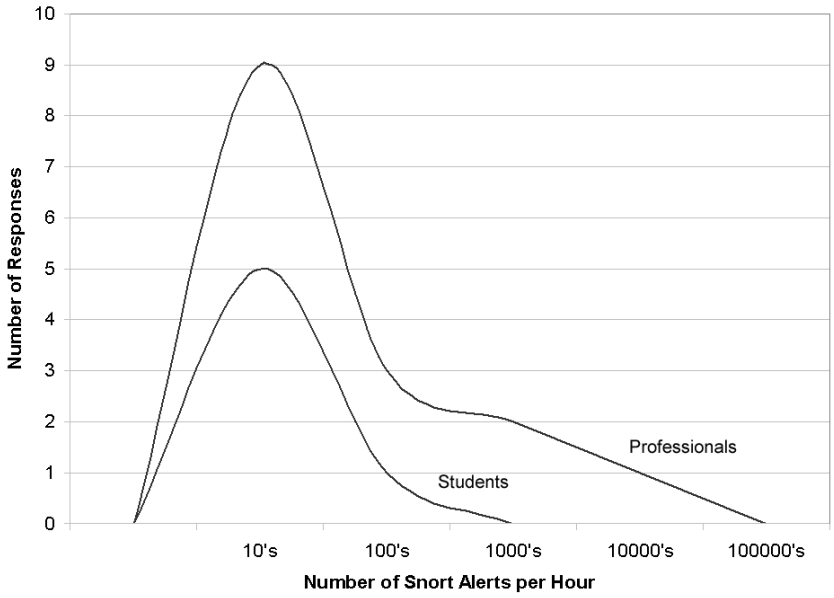
course of conducting the survey indicated that there was some confusion and we consider our results inconclusive. We believe these questions to be important and will more precisely word them in the future.

Analyst Overload

The second section of the survey asked the participants to estimate the point at which they, and not their given computing platform, became overwhelmed when using Ethereal and Snort. To prevent confusion, the survey included specific instructions that clearly asked participants to consider only human, and not machine, performance. The results are shown in Figure 6.1. The figure shows that both professionals and students have a distinct tipping point when they become overwhelmed with security data. We did exclude two outliers from our Ethereal results. These values, one million packets and one hundred thousand packets, are shown in the figure, but were removed in our statistics below. Using Ethereal students became overwhelmed when faced with datasets containing an average of 635 packets for students (n=7) and professionals became overwhelmed with 5,905 packets (n=18). While these statistics are based on self-reported data, we believe it is a reasonable assumption that information security students and, in particular, professional security practitioners, would be able to estimate this value. When comparing the averages for both groups it is important to note that professionals reported overload at approximately an order of magnitude higher than students. Figure 6.1b illustrates a similar tipping point for Snort alerts. When asked how many Snort alerts they could handle per hour, students reported reaching overload at an average of 30 (n=6, standard deviation = 35.70) and professionals were able to handle an average of 1,183 alerts (n=15, standard deviation = 2,779.70). Again, the professional group



(a) Ethereal overload



(b) Snort overload

Figure 6.1: Analyst overload as reported by security professionals and information security students.

reported a significant improvement over students. Based on informal discussions with

the professional group, we believe the primary reason was an intimate knowledge of the networks under their care as well as their use of reporting tools, backend databases and custom processing scripts. The remaining questions in this section of the survey sought to clarify how easily attackers could worsen these tipping points by the insertion of malicious data designed to overwhelm or mislead analysts. These questions are related to, but distinct from, those shown in Table 6.1.

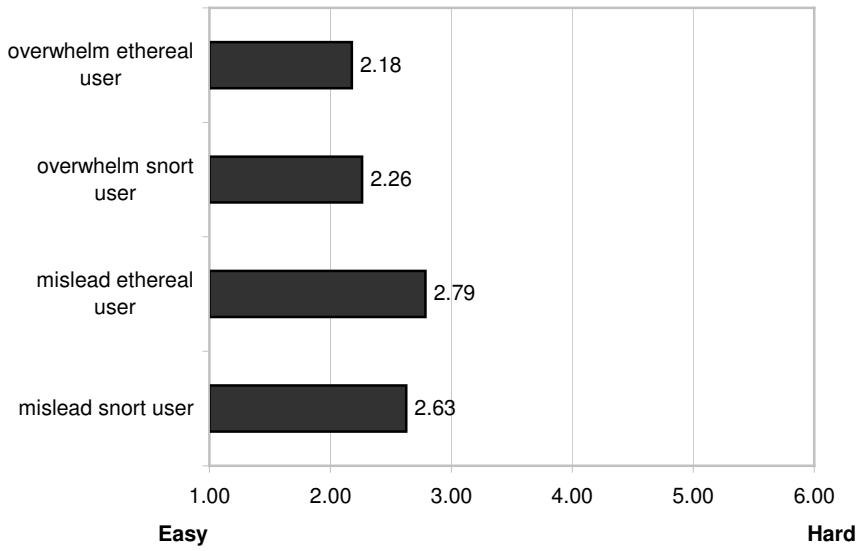


Figure 6.2: Ability to confuse and mislead Snort and Ethereal users

The results in Table 6.1, show how difficult it would be for an attacker to insert malicious data into security related datasets. These questions deal specifically with the impact malicious data would have on user of the Ethereal and Snort applications. Our results, shown in Figure 6.2, show that the survey participants believed that both Ethereal and Snort were susceptible to denial of information attacks against users of these tools. On a Likert scale (1=easy, 6=hard) overwhelming users was an average of 2.18 (n=28, standard deviation = 1.219) for Ethereal and 2.26 (n=23, standard deviation = 1.176) for

Snort. While we did not achieve consensus on the reason for a lower average for Ethereal, we believe it was likely due to its limited textual display and tree based GUI. This assessment is supported by our findings presented in Table 6.2. In discussions with participants following the survey, there was an informal consensus that denial of service attacks against the tools would be easier to technically implement, misleading attacks would require more subtlety and skill on part of the attacker. Overwhelming users of both Ethereal and Snort was perceived as being noticeably easier than misleading them. Snort users were slightly easier to mislead with an average score of 2.63 (n=24, standard deviation = 1.527). Ethereal received an average score of 2.79 (n=28, standard deviation = 1.475). We believe Snort was deemed as more susceptible because of its vulnerability to arbitrary alert triggering attacks described in [87].

Snort and Ethereal Strengths and Weaknesses

The third section of the survey aimed to clarify why this overload occurred by identifying strengths and weaknesses of both Ethereal and Snort. This section of the survey included free form fields that we have summarized in Table 6.2. Of the 39 survey participants, 32 included at least one strength or weakness in their response. The left column of the table indicates the strengths and weaknesses and the right column shows the percentage of survey participants who included this comment. Ethereal's primary strengths included the ability to analyze protocols and drill down to details of individual packet parameters, but was hampered by a overwhelming detail, loss of big picture context and a cumbersome user interface. Snort was highly regarded for its high quality signature database and robust filtering language, but was criticized for the number of false positives, reliance on

and the complexity of known signatures as well as poor quality front ends. While these are valid concerns and would be a valuable topic for future research, in this paper we will focus specifically on addressing the shortcomings of Ethereal's packet level analysis using our visualization tool. In particular, the following section will evaluate these shortcomings using two human-centric experiments.

Overall, we were pleased with the results of our survey and believe it helped to answer the questions we sought to address. Participants were largely from our target user audience of professional security analysts, which is typically an elusive population to gain access to in large numbers. We believe participants were experienced enough to effectively answer the questions we posed in the survey. There was a clear consensus that it is easy to insert malicious content into intrusion detection alerts, network packet captures and system logs. Participants believed that it is straightforward to overwhelm users of Snort and Ethereal and only slightly more difficult to mislead them. We helped clarify the threshold at which Snort and Ethereal become difficult to use. While the threshold was higher for professional security practitioners than for information security students, it was still quite low when considering the ability of attackers to insert malicious data into their datasets. Finally, the survey participants provided a very detailed set of strengths and weaknesses for both tools. Based on these results it is clearly possible to overwhelm or mislead users of these two widely deployed tools. The system we designed, RUMINT, was created to address these issues, specifically those of packet level analysis and is evaluated in the following section.

Table 6.2: Summary of Ethereal and Snort Strengths and Weaknesses

Ethereal Strengths (30 Respondees)	Percentage of Responses
Full view of all packet parameters	27%
Capture and display filters	27%
Dissect and analyze protocols	27%
Ability to sort list elements	7%
Summary by packet type/packet headers	7%
Network stream reconstruction	7%
Ability to use coloring rules to highlight packets	7%
Ability to view packets in hexadecimal	3%
Detail oriented search for something specific	3%
Various modes of capture (live and to disk)	3%
Interface	3%
Statistical analysis of data (RTT times)	3%
Ability to filter network streams	3%
Ability to alert on known signatures	3%
Great for beginners	3%
Great deep packet analysis down to the bit	3%
Human readable packet contents / ASCII view	3%
Great for performing spot analysis	3%
Troubleshooting misconfigured networks	3%
Open source	3%
Filtering	3%
Flexibility	3%
Ethereal Weaknesses (27 Respondees)	**Percentage of Responses**
Overwhelming detail / too much for human to process	22%
Impossible to properly visualize a large dataset without getting lost and confused	11%
GUI too cumbersome	11%
Complex filter syntax	7%
Interface does not scale well	7%
Loss of context / big picture is hard to see	7%
Time consuming	7%
Small/limited data views	7%
Difficult to find patterns and trends	7%
Too much information for a list representation	7%
Not good for constant analysis over a long period of time	4%
SSL makes much of the payload uninformative	4%
Not good at network flow analysis	4%
Inability to disregard non-relevant packets	4%
Lack of time based histograms for given hosts	4%
S/N ratio is low in a busy network	4%
Snort Strengths (18 Respondees)	**Percentage of Responses**
Robust and configurable filtering	39%
High quality signature database	28%
Helps to focus human resources	11%
Flexibility	11%
Ability to access details of packets/alerts	11%
Open source	11%
Useful as a packet capture program with signature matching capability	6%
Quick and dirty intrusion detection	6%
Trend analysis	6%
De-facto standard	6%
Automated analysis of data over a long period of time	6%
Flow based approach	6%
Cost	6%
Snort Weaknesses (17 Respondees)	**Percentage of Responses**
Too many false positives	41%
Reliance on known signatures	41%
Time and difficulty in selecting right set of signatures for a given network.	18%
Front end GUIs are poor	12%
Tedious analysis	6%
Dependency on well written rules	6%
Steep learning curve	6%
Snort management	6%
Tuning rules too tightly will cause false negatives	6%
Lack of good intrusion detection rules	6%
Backend database slows down with several million entries (MySQL)	6%
Lack of visual (e.g. non-text) output	6%

Quantitative Security Evaluation Design and Results

While the survey identified clear problems associated with both Ethereal and Snort, in this section we focus specifically on evaluating RUMINT's effectiveness in addressing the key weaknesses of Ethereal: overwhelming detail, lack of high-level context and a cumbersome graphical user interface. Specifically we sought to determine if human analysts can visually detect and precisely locate malicious activity in network packet datasets. We also attempted to determine how long it takes an analyst to perform these activities as well as which classes of malicious activity were most easily detectable. In addition, we collected initial statistics which provide insight into how much task specific training and analyst experience is required to accomplish these tasks. Finally, we sought to determine how rapidly analyst performance improves during the course of using the tool. In the experiments, tasks were embedded into four questions: packet header characterization (Q1) as well as detecting and precisely locating a random data anomaly (Q2), fragmentation anomaly (Q3) and 0x90 anomaly (Q4).

Evaluation Design

With the exception of Question 1, both experiments required participants to perform the same tasks. The first experiment was conducted with 10 students participating in a Master of Science in Information Security graduate course. They completed the entire set of tasks in a laboratory running RUMINT, a Windows XP application, inside of a VMWare virtual machine. The host systems were 1GHz machines running Red Hat

Enterprise Linux. The second experiment was conducted with 22 students in a 4000 level course in network security. Before beginning work on the questions, both groups were given a short presentation on security visualization and general use of RUMINT by one of the experiment coordinators. After the presentation, students were given 15 minutes to practice with RUMINT using a sample packet capture. At the end of the hands-on practice, participants were given the testing dataset and the question worksheet. Participants in the second group completed the hands-on portions of the experiment using their own personal computers as a take-home exercise. Both groups used version 1.94 of RUMINT, available here [91], for the evaluation. Response times were self-reported using the operating system's clock. Table 6.3 summarizes the characteristics of both experiments.

Table 6.3: Summary of Experiment Construction

Experiment	Group	Participants	Training Time	Practice Time	Questions	Notes
1	MS Infosec Students	10	30 minutes	15 minutes	Q1, Q2, Q3, Q4	Conducted in laboratory.
2	BS EE Students	22	45 minutes	15 minutes	Q2, Q3, Q4	Take home

Metrics

During the design of our experiments, we considered a wide variety of metrics to evaluate the performance of our system: task completion, time to completion, error rate, satisfaction, training time, learnability, networking and computer security skill level, adoption by others, website downloads of RUMINT, frequency and duration of use in operational environments, perceived control and perceived ability to complement other tools. Ultimately we settled on time of completion and error rate as our primary metrics

in order to reasonably scope the experiments.

Tasks

The tasks we chose were designed to evaluate the performance of RUMINT in the execution of a common analytic task: rapidly gain context on an unfamiliar set of network packets and identify and precisely locate suspicious activity for deeper analysis. We chose these tasks because lack of context was a key shortcoming identified in our survey of analysts. Ethereal is the tool of choice for packet level analysis, but it quickly became overwhelmed with a relatively small number of packets. These tasks were embedded in four questions in the following manner:

Question 1 (Q1): Header Characterization

- Characterize 19 header fields into one of four categories: constant, small number, medium number and large number.

Question 2 (Q2): Random data anomaly

- identify what packet number the sequence begins and ends
- identify the number of packets in the sequence
- identify the destination port
- characterize the byte composition of the suspicious packets as primarily printable ASCII or across the entire possible range (0-255)

Question 3 (Q3): Fragmentation anomaly

- Precisely locate the anomaly by determining the start and end packet numbers.
- Identify the two highest and two lowest values of the fragmentation field for this anomaly

Question 4 (Q4): 0x90 Anomalies

- Precisely locate the anomaly by determining the start and end packet numbers for the two groups of packets containing a large number of 0x90 values.

Table 6.4: Comparison of Potential Dataset Sources

Source	Noise Level	Reflection of Reality	Publicly available	Notes
DEFCON Capture the Flag	Low	Good	Yes	Multiple teams attack and defend networks in an isolated environment.
U.S. Military Cyber Defense Exercise	Low	Good	Yes	A National Security Agency red team attacks networks defended by students.
Honeynet Project Scan of the Month Competition	Low	Good	Yes	An online competition based on live honeynet captures. These datasets are well studied and analyst solutions are available online.
Georgia Tech Honeynet	Low	Very Good	No	Live capture from the Georgia Tech Honeynet.
Georgia Tech Campus Network	Very High	Excellent	No	Live capture from the Georgia Tech campus network.
Custom Dataset created for experiment	Variable	Poor	Yes	Creation of a dataset in a laboratory environment.
User Selected	Variable	Excellent	No	Provided by user based on real-world requirements. Privacy concerns likely.

Dataset

To support our experiments, we examined a number of sources of network datasets for possible inclusion: Defcon Capture the Flag [94], United States inter-service academy cyber defense (CDX) exercise [116], Honeynet Project Scan of the Month competition [56], captures from our university honeynet [83] and real traffic from our university campus network. We also considered creating a custom dataset and allowing individual participants to provide their own real-world datasets. Table 6.4 summarizes each of these options. Ultimately we selected the U.S. military's CDX dataset because it was publicly available and reflected live malicious activity without substantial noise of legitimate traffic. Clearly, coping with varying degrees of legitimate traffic is of concern for security visualization systems, but we chose to leave exploration of this topic for future work. We also believe that the Honeynet Scan of the Month competitions are well worth evaluating in the future as they contain live traffic capture from the Honeynet Project's network of sensors and are well studied. Multiple analyses of each dataset are available online. The CDX dataset consists of 919 packets and contains a number of interesting anomalies. These anomalies form the basis of our four questions (Q1-Q4) and allowed us to test the experiment participants' ability to identify and precisely locate them.

In conjunction with our human-centric evaluation, we also sought out a dataset that was not readily analyzed with automated approaches. We believe visualization is best applied to security problems that cannot readily be addressed using automated solutions. Using visualization, humans can inherently detect different patterns and anomalies that are difficult or impossible using machine processing. Machines are best at finding known patterns and signatures, but are weak when tasked with detecting abnormalities. In short, visualization is at its best when you do not know what you are looking for. An example

of this is the creation of pattern matching rules for signature based intrusion detection systems such as Snort. We ran the CDX dataset through Snort version 2.4.3 using the official Sourcefire Vulnerability Research Team rule set version 2.4 released in April 2005. Snort detected only an NMAP XMAS scan and generated 87 associated alerts, but nothing else. While we applied the default rule set which most security professionals would fine tune to meet their individual needs, Snort's performance does underscore the need for visualization and support the use of the CDX dataset in our evaluation.

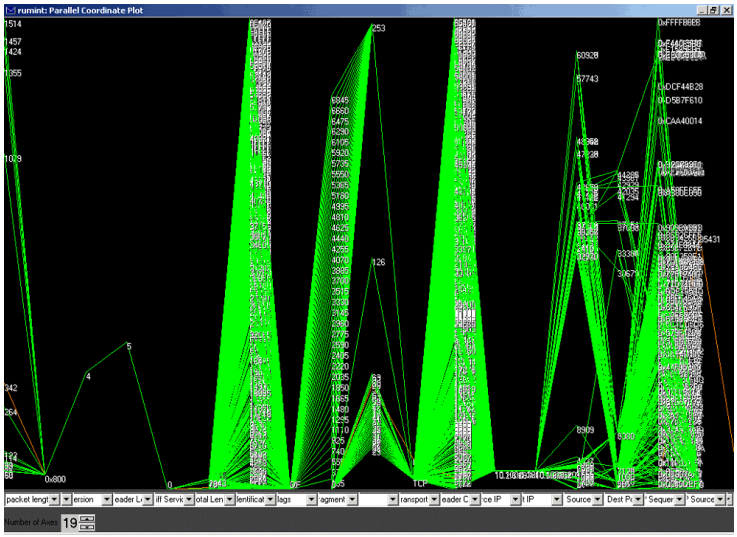


Figure 6.3: Visualization of 19 header fields using a parallel coordinate plot. While this snapshot shows the header fields from the entire 919 packet dataset, study participants could use the PVR interface to selectively play back and step through traffic from any point.

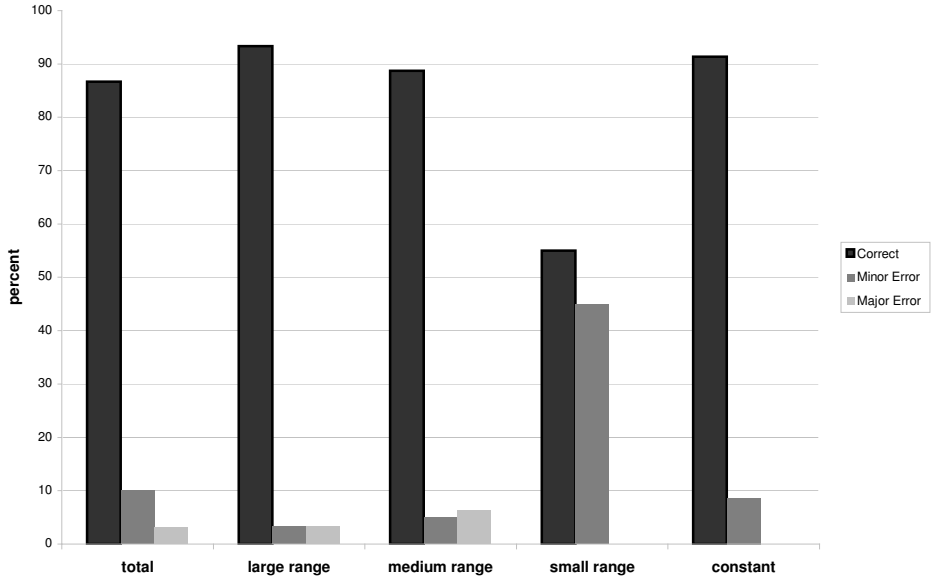


Figure 6.4: Results from characterization of header fields task (Q1).

Evaluation Results

Rapidly Characterize Packet Header Fields (Q1)

A key component of gaining context of a new packet capture is acquiring a rapid understanding of the packet header fields. The behavior of header fields gives insight into potential malicious behavior. RUMINT supports visualization of 19 header fields: packet length, ethertype, IP version, IP header length, IP differential services, IP total length, IP identification, IP flags, IP fragmentation offset, IP time to live (TTL), IP transport protocol, IP header checksum, source IP address, destination IP address, TCP source port, TCP destination port, TCP sequence number, UDP source port and UDP destination port. Participants were instructed to play back the dataset and observe the results using its parallel coordinate plot visualization window. Study participants were then asked to rapidly characterize these 19 header fields for all packets in the dataset into four categories: constant, small number of values, medium number of values and large number of values. The training phase of the experiment did not include any specific training on this task. Figure 6.3 shows the visualization window after packets 1-919 were played back using the PVR interface. Students were able to use the interface to begin playback at any position in the dataset as well as step through the traffic. Results from this task are shown in Figure 6.4. The figure depicts correct answers as well as major and minor errors. Minor errors indicate an answer that was one size category away from the correct answer, e.g. selecting constant when there were actually a small range of header values. A major error indicates an answer that was two or more categories distant from the correct solution.

Study participants were able to characterize header fields with a high degree of

accuracy. The average completion time for this task was 8.03 minutes for the 10 students (standard deviation = 3.31). The overall success rate 86.71%. The participants performed well at identifying large (93.33%) and medium ranges of values (88.75%) as well as constants (91.38%), but ran into difficulty when distinguishing between a small range of values and constants. The dataset contained a number of fields that contained only two closely placed values. The label of the second value occluded the label of the first value and participants were only 55.00% successful. While it was still possible to observe the difference as the traffic was played back, careful attention was required. Overall, we were satisfied with the results of this stage of the experiment and decided to omit the question in the second experiment. We used the additional time to provide extra training time for this less experienced group. Based on post experiment discussions with the participants, we project that significant increases in efficiency would be possible on future iterations of this task.

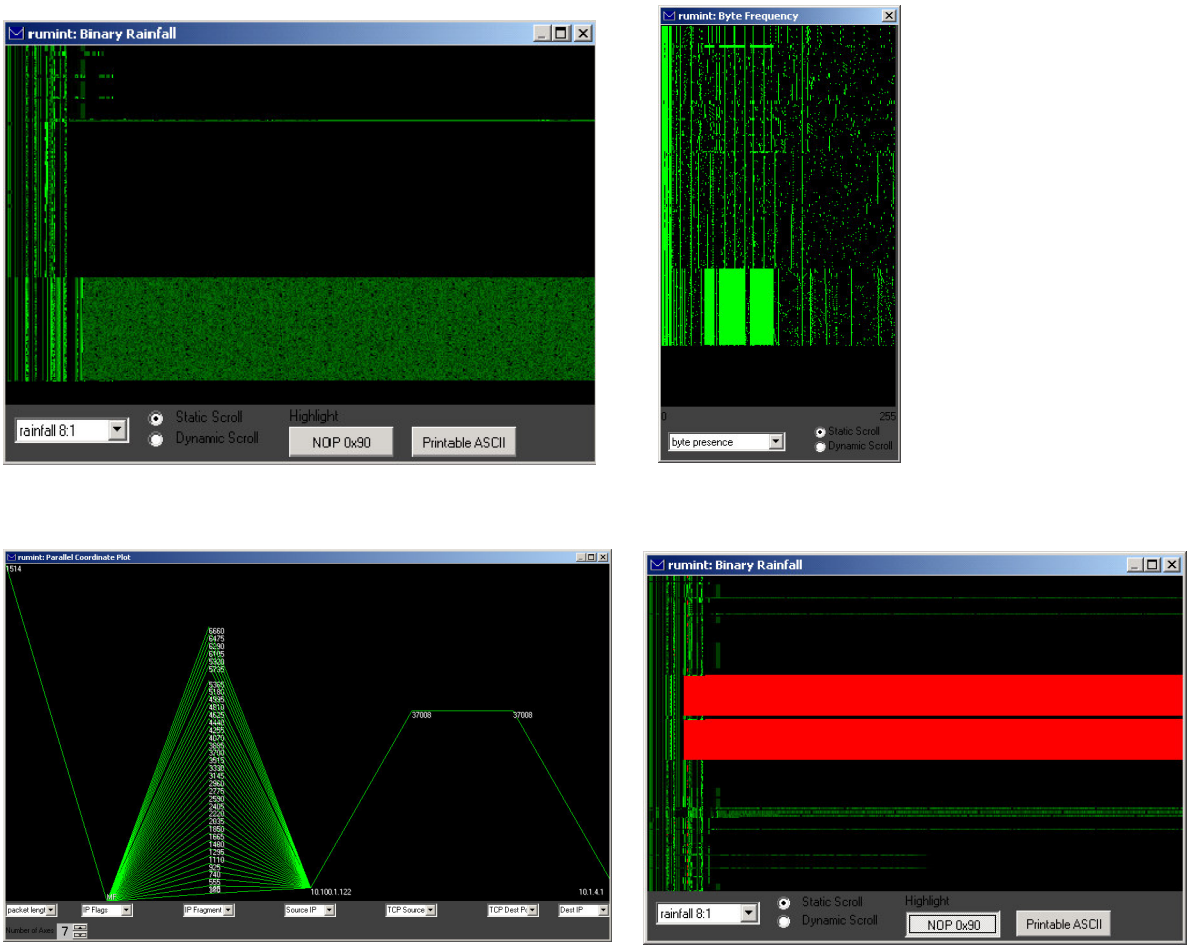


Figure 6.5: Visualization snapshots of the Q2-Q4 anomalies. Images include the random data anomaly (top left), fragmentation anomaly (bottom left) and 0x90 anomaly (bottom right). The top right image shows the byte frequency distribution of the random data anomaly.

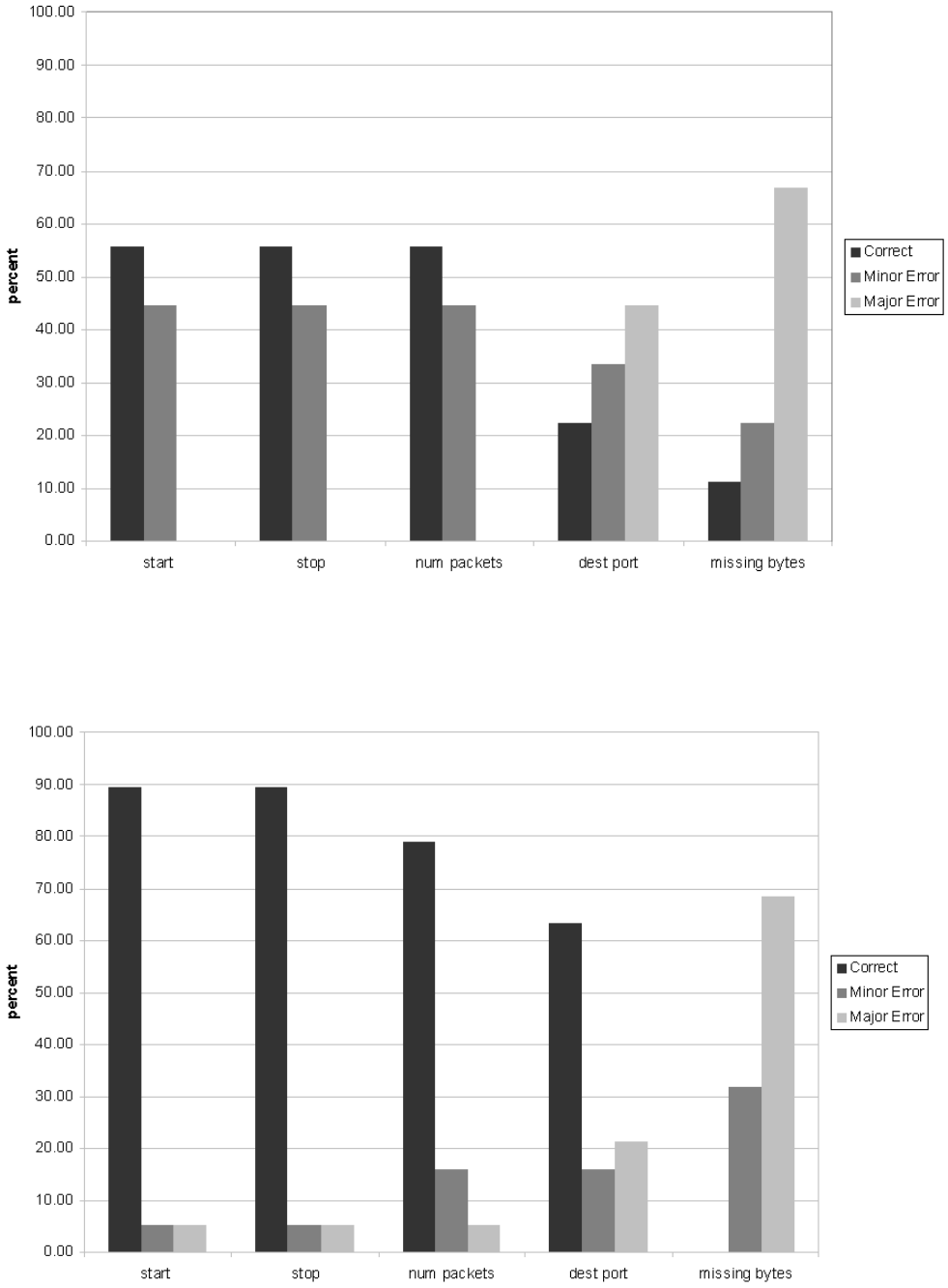


Figure 6.6: Random data anomaly results (Q2): Study participants were tasked to locate the start and stop positions of the anomaly as well as determine the number of packets, destination port and bytes missing from each of the packets. The top chart shows results from the graduate student group. With slightly more training time results improved significantly for the undergraduate student group as seen in the bottom chart.

Question 2 (Q2): Random Data Anomaly

Random data within network traffic can be indicative of covert channels or trojaned hosts, but rapidly detecting and analyzing such traffic can be difficult. To test the efficacy of performing this task we asked participants to locate "a sequence of more than 50 large packets that contain seemingly random data." They were told to use the binary rainfall and byte frequency visualizations provided by RUMINT. Both techniques are described in Chapter 5. To summarize, the binary rainfall view plots packet contents, one byte per pixel, with one packet per horizontal line. Pixels are mapped to the bytes comprising the packet. Subsequent packets are plotted on the next lower line. Large packets will consume most of the horizontal line and random payloads will appear as white noise. This region can be seen in the top left image of Figure 6.5. The byte frequency visualization also plots packets one per line, but illuminates pixels based on the presence or frequency of a given byte in a packet. For example, if packet 10 contained bytes 20 and 30, pixels (20,10) and (30,10) would be illuminated. Both visualization techniques wrap around to the top of the window when the display is filled. After locating the anomaly, participants were asked to determine the start and end packet numbers, the number of packets in the sequence, the destination port and to determine which bytes were not used in these packets.

The graduate group met with mixed success as they attempted this first anomaly detection task with a 55.56% overall success rate (Figure 6.6, top). Of this group, 44.44% mistook the 0x90 anomaly (Figure 6.5, bottom right) for the random data anomaly (Figure 6.5, top left). They two anomalies are similar visually, but the 0x90 anomaly contained payloads of identical bytes. After analysis of the graduate group

experiment, we included a short discussion on the visual characteristics of random data in our training session for the undergraduate group using a practice dataset. As seen in Figure 6.6 (bottom), 89.47% of the undergraduate group correctly identified the start and stop positions of the anomaly and only 5.26% confused it with the 0x90 anomaly. Even when an anomaly was incorrectly detected, both groups were successful in identifying the start and stop position of the anomaly they identified, with a slightly lower performance shown by the undergraduate group. Both groups were less successful at identifying the destination port in use for the anomaly. This task required the participants to use both the binary rainfall and parallel coordinate plot visualizations to identify the destination port in use and we believe this was the cause for the decreased performance. Similar to the port identification task, the participants were also asked to combine the use of the binary rainfall visualization and the byte frequency display to identify which bytes were missing from the apparently random sequence of data. Specifically, successful completion of the task required the students to mouseover regions of the byte frequency display (Figure 6.5, top right). The graduate group only succeeded 11.11% of the time and no one from the undergraduate group was successful, despite increased coverage of the task in the undergraduate training session. Informal discussions with study participants indicated the cause to be a lack of a zoom function in the byte frequency visualization and the general lack of understanding concerning packet level byte distributions. We plan to add a zoom capability to the tool visualization window and readdress this topic at a later date.

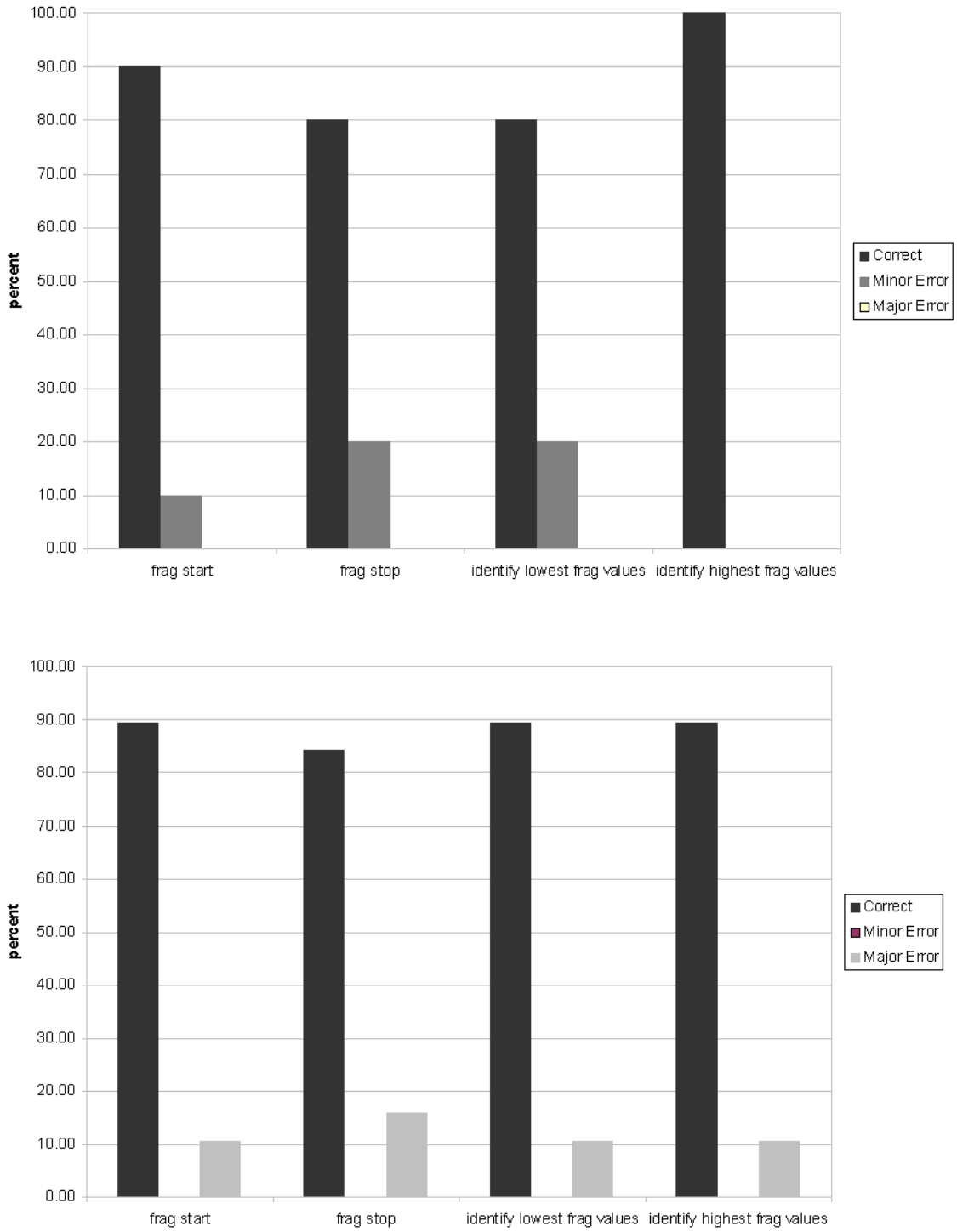


Figure 6.7: Fragmentation Anomaly Results (Q3): Study participants were tasked to locate the start and stop positions of the fragmentation anomaly and determine the highest and lowest fragmentation field values. The top chart shows results from the graduate group and the bottom chart is the undergraduate group.

Question 3 (Q3): Fragmentation Anomaly

Fragmentation attacks are popular techniques for bypassing intrusion detection systems. This question asked participants to find the "significant fragmentation anomaly" within the dataset. They were told to use the parallel coordinate plot display and ensure that one of the vertical axes was set for the IP fragmentation field. No further guidance was given. They were asked to determine the start and end packet number of the anomaly and to determine the two highest and two lowest values of the fragmentation field.

When analyzing the results from this question, we noted that the performance of the graduate group is now on par with the undergraduate group despite the advantage of additional training. The graduate group correctly identified the start and stop positions of the anomaly 90 and 80 percent of the time, respectively. The undergraduate group succeeded at 89.47% and 84.27% for the same tasks. We attribute this increase to the deeper level of expertise of the graduate students combined with the hands on experience garnered during questions Q1 and Q2. Likewise, both groups improved their use of the binary rainfall visualization in conjunction with the parallel coordinate plot and succeed at identifying the range of fragmentation values approximately 90% of the time.

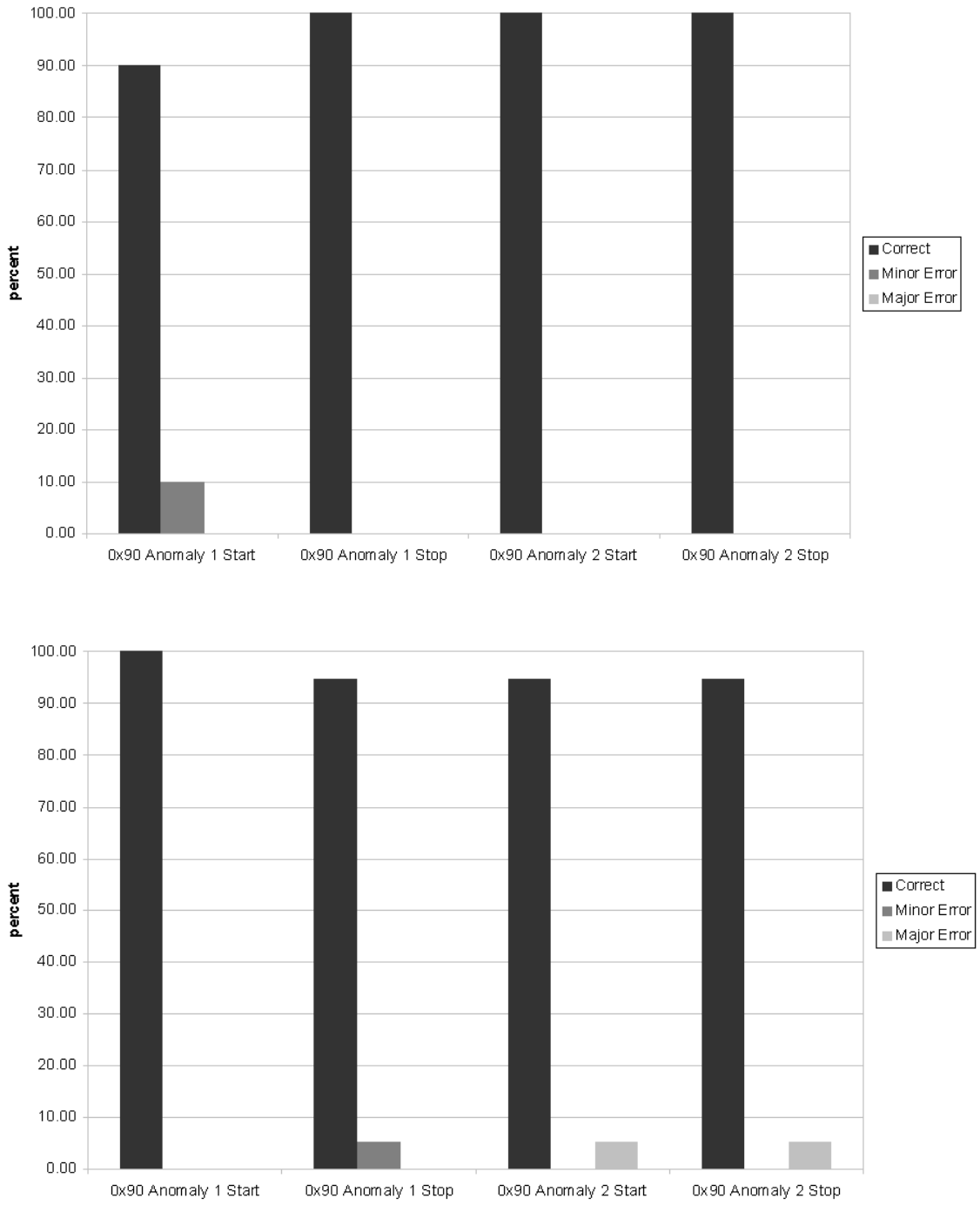


Figure 6.8: 0x90 Anomaly Results (Q4): Study participants were tasked to locate the start and stop positions of the two anomalies consisting of packets containing a large number of 0x90 values. The top chart shows results from the graduate group and the bottom chart is the undergraduate group.

Question 4 (Q4): Two 0x90 Anomalies

Buffer overflows are a major problem in today's network security environment. Many implementations of this attack use a no operation (NOP) sled to allow the attacker a larger target to direct code execution. Traditional buffer overflow attacks use a series of 0x90 values as a NOP sled because 0x90 equates to the no operation instruction on Intel CPU's. Because of this use, we included a button on the binary rainfall display that users display all packet contents of value 0x90 in bright red. More recent variations of the buffer overflow use sequences of idempotent instructions that are equivalent to NOPs in order to hide their presence. We plan to explore these more advanced obfuscation techniques in the future, but 0x90 NOP sleds are still in use and we believe this is a valuable first step in the human detection of buffer overflows. Participants were instructed to find the two 0x90 anomalies (shown in red in the bottom right image of Figure 6.5) within the dataset and record their start and end packet numbers.

The results from question 4 show a similar increase on those from question 3. Both groups succeeded at the 0x90 detection task approximately 95% of the time with nearly identical results between the two groups. Similar to the increase in successful task completion with each subsequent anomaly detection task, there was a distinct decrease in task completion time (Figure 6.9). It took the graduate group, on average, 16.22 minutes (standard deviation = 6.22), 5.16 minutes (standard deviation = 1.53) and 2.90 (standard deviation = 1.45) minutes to complete tasks Q2-Q4 respectively. While each question contained a varying number of tasks and therefore cannot say conclusively, we believe that this significant decrease in task completion time was based on increasing familiarity with the tool.

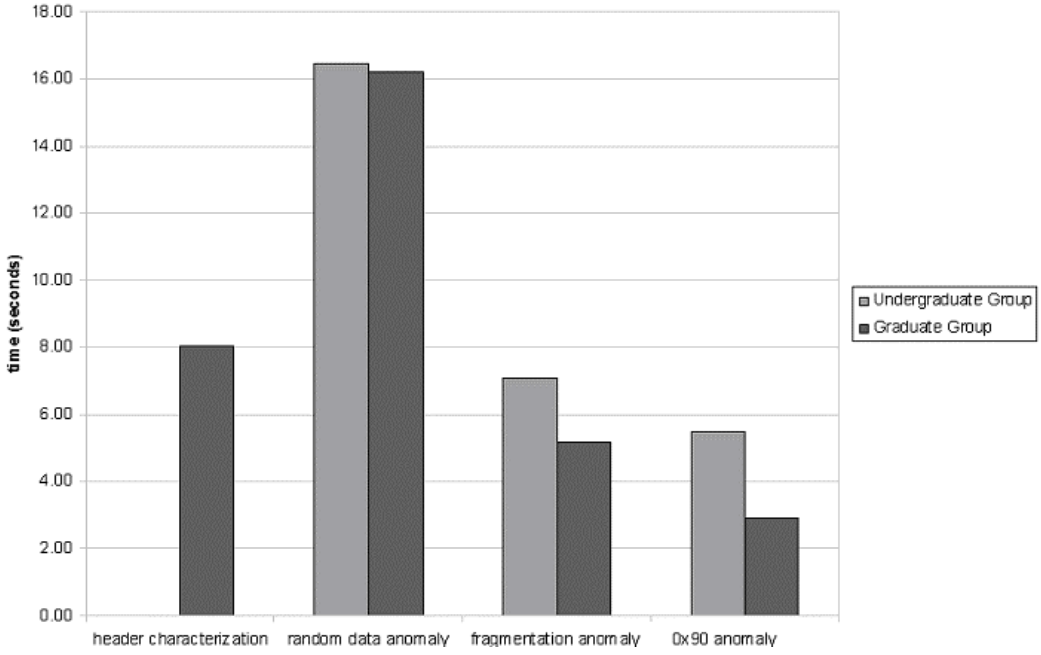


Figure 6.9: Task Completion Time Results. Both groups experienced significant decreases in task completion on the anomaly detection tasks.

After the quantitative portions of the evaluation, we asked participants to estimate the point at which overload occurs using RUMINT. As none of the study participants completed our survey reported in the previous section, we also asked them to evaluate the overload point of Ethereal. We combined these new Ethereal responses with the student responses from our initial survey and plotted them in conjunction with RUMINT responses on Figure 6.10. The combined student groups yielded a total of 29 Ethereal and 15 RUMINT responses with average overload occurring at 907.79 packets (standard deviation = 1,806.80) for Ethereal and 9,056 packets (standard deviation 13,434.96) for RUMINT. Based on this qualitative assessment we achieved about an order of magnitude better performance than Ethereal.

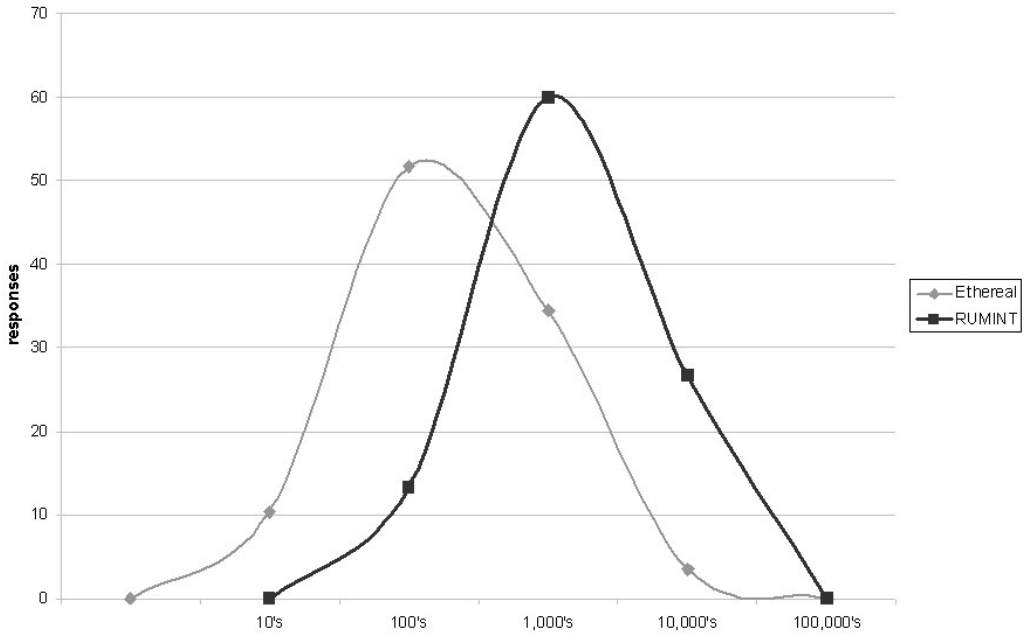


Figure 6.10: Comparison of overload point of Ethereal and RUMINT based on a percentage of responses. RUMINT demonstrated approximately an order of magnitude better performance when compared to Ethereal.

In summary, we believe RUMINT was successful in reducing denial of information problems for security analysts. Humans were able to detect malicious activity in the playback of forensic network traffic. The malicious activity we tested in questions 2-4 were large visual anomalies within the dataset and therefore, the full extent of visual anomaly detection for security visualization still remains a ripe area for future work. When anomalies were detected, participants were able to precisely locate the packets of interest due to the strength of the PVR interface metaphor. We believe this capability is critical to facilitate additional follow-on analysis with a tool such as Ethereal. While the

PVR interface was successful, the addition of packet identification capabilities to the individual visualization windows, e.g. pop up windows generated by placing the mouse pointer over a packet, would be a useful addition. Our results indicate that only limited training on security visualization and use of the tool was required for effective use by individuals knowledgeable in network security. In our experiments, participants were given 30-45 minutes of overview training and were allowed 15 minutes for hands on practice. Despite the short training period, they rapidly grasped the usage and application of RUMINT. In addition, task completion time dropped dramatically during the evaluation session from 16 minutes on Q1 to 5 minutes on Q2 and 3 minutes on Q3 further supporting our assessment. We feel that the basic visual security analysis approach of RUMINT is sound, but would be enhanced by future work in several areas: the creation of a library of visual snapshots of legitimate and malicious activity, adding the ability to easily zoom into the visualization windows and by increasing filtering and color coding capabilities. We believe our general evaluation methodology approach was sound, but would like to further evaluate the tool with professional security analysts in their normal work environment. Study materials from the survey and experiments, as well as the packet captures used, are available online [18].

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Denial of information is a problem that we face in a wide variety of areas. In particular, the adversarial nature of the computer security domain illustrates the constant tension between malicious entities and the operators charged with defending an organization's information resources. While there are many possible solutions, we have explored the use of carefully crafted information visualizations to cope with the massive, and possibly tainted, amount of security data generated by network sensors and host-based applications. Previous techniques such as manually traversing textual logs, using command line analysis scripts and traditional graphing and charting techniques provide only a limited defense against overwhelming or misleading human defenders. With these existing tools it is difficult to gain a coherent picture of network health and security status. In many instances, this flood of data will actually reduce the overall level of security by consuming the operator's available time or misdirecting their efforts. In extreme circumstances, the operators will become desensitized and ignore security warnings altogether, effectively negating the value of their security systems. Information visualization of security related data bears great promise in countering these problems and, ultimately, will assist in making our personal computers, servers and networks more secure. Such work is both an art and a science requiring expertise from the computer graphics, information visualization, interface design and security communities in order to turn the raw security data into insightful and actionable information and knowledge.

With the problem of denial of information there is no shortage of raw data, in fact there is far more than can be analyzed without employing more advanced techniques.

Information Visualization is one way of effectively communicating information. Deception is one way to negatively affect this capability. Today's systems are being used in critical applications to glean insights that are difficult to see using traditional non-visual techniques. As malicious entities become aware of the power of these tools, the tools themselves and the decision makers that use them will increasingly become the subject of attack. These vulnerabilities may manifest as significant attacks and we have provided real world examples to show that these attacks are real. Any system that uses data from malicious trusted or untrusted sources is at risk. Today's visualization technology has not been designed with consideration of these risks and the notion of active malicious entities. Even carefully user-customized applications are vulnerable due to incorrect defaults, limitations in the visualizations themselves and weaknesses in the overall system. To better defend information visualization systems against denial of information attack we have presented a framework and taxonomy for analysis, presented viable attacks from the network security domain as well as developed design principles and assumptions to help create systems that protect both the system and the user. We used these principles to design an attack resistant tool, RUMINT, that is capable of both real-time and forensic analysis of packet-level data. The system is capable of replaying large capture files and providing much needed insight and big picture context.

The system has proven to be a useful tool for multiple purposes. We effectively used it in a wide variety of instances to get a rapid overview of datasets. This was a crucial aspect when analyzing our three-year honeynet capture archive of approximately 100 GB.

Especially for these forensic analysis tasks, we found a significant decrease in the time required to find points of interest in the capture files.

RUMINT can give a much faster high-level impression about traffic patterns and events than an analysis with Ethereal or similar programs. Once areas of interest have been identified in the data using our visualization tool, Ethereal can then be used to do a deep-protocol analysis. We believe that incorporating similar protocol analysis directly into our tool is both technically feasible and a logical direction for the future.

As part of the design and testing of RUMINT we explored the combination of dynamic queries, semantic zooming and interactive encoding with a visualization technique for comparing large numbers of binary objects. We then used the system to study several datasets containing large numbers of malicious and non-malicious network packets. We believe the binary rainfall visualization technique is useful for off-line forensic analysis of network datasets and, to a lesser degree, for real time network monitoring and intrusion detection. In both applications it should be augmented with dynamic queries, semantic zooming and interactive filtering to eliminate noise and highlight areas of interest. We also believe that these techniques would work extremely well for navigation within and analysis of binary files, particularly when combined with semantic encoding and transforms based on knowledge of the file type.

Our fingerprinting results demonstrated that common protocols as well as popular attack tools of the network reconnaissance and vulnerability assessment classes can be readily detected by passive promiscuous mode sniffing and appropriate visualizations. While some occlusion occurs, fingerprints are frequently visible despite the visual noise of routine traffic. For future work we envision fingerprinting worm behavior and other

well-known malicious network activity (e.g. spyware, trojans and warez servers), dual-use activities (e.g. FTP servers, Internet Relay Chat servers) as well as typically legitimate network activities (e.g. email, web browsing). Our long-term goal is to create a library of visual signatures that can be used by the expert or novice analysts to detect malicious activity. Ultimately, we believe that visual intrusion detection systems can effectively supplement traditional signature and anomaly based intrusion detection systems, but care must be taken avoid overwhelming the human operator. To this end, the effectiveness of any tool is defined by its usability. Careful task-driven usability studies that optimize ease of use and analyst intuition, will make visual intrusion detection systems more successful and drive down the skill barrier required for effective use.

We believe our evaluation of RUMINT was successful, in that we gained significant insight into its ability to counter denial of information attacks. Our survey confirmed our intuition that many sources of security data are susceptible to corruption by attackers, that the overload point at which human users of two extremely popular analysis tools, Ethereal and Snort, is relatively low and that these tools are vulnerable to attacks which attempt to mislead or confuse the user. The survey also provided a large number of strengths and weaknesses that are useful to design better tools. By designing RUMINT to complement the weaknesses of Ethereal we then focused on evaluating several tasks that examined the analyst's ability to visually detect and precisely locate anomalies within unfamiliar network packet captures. In general, our study participants were successful and their informal feedback was largely positive, particularly with regard to the potential of visual approaches to the analysis of security data. Our experiments

provided lessons learned that we believe will be useful by other researchers attempting to evaluate their security tools as well as quantitative results that demonstrated a rapid increase in performance with a small increase in training time and hands on use of the tool. Additionally, our results indicate that information security students using Ethereal reach overload at approximately 908 packets and users of RUMINT overload at 9,057. Our tool does not match the deep protocol analysis capabilities of Ethereal, but we do believe that it provides about an order of magnitude better performance when attempting to gain context of network packet captures. For future work we plan to incorporate feedback into the design of RUMINT by adding the ability to combine multiple security related data sources into a combined display as well as improve the RUMINT's interactive filtering and reporting capabilities. We also plan to continue to engage the professional computer security community to gather requirements for our future systems as we believe this step is critical to produce relevant work. Finally, we plan to further quantify the threshold at which overload occurs, particularly as malicious data is introduced into the datasets under analysis.

There are many ripe areas for future work in the area of denial of information and security visualization. When trying to determine these areas it is useful to reexamine the framework for security visualization systems presented in Figure 5.1. For example, each visualization system is only as good as the data upon which it relies. Today's systems utilize only one or two security data flows, but there lies great potential in the combined use of many of the possible data streams from existing security solutions. Examples include: packet captures, intrusion detection/prevention system logs, syslog, firewall logs, anti-virus reports, netflows captures, routing data, client/server process listings,

honeynet logs and data generated by other network appliances. Allowing the user to interact with as many possible security data streams as possible is a sound beginning, but will likely result in too much information without advances in filtering techniques. As a starting point, Ethereal's comprehensive command line filtering language is a good example to emulate and can be extended to all possible data streams, but it should be complemented with carefully designed graphical user interface components. These interfaces components will allow the user to manage the complexity of multiple data streams and could be used in parallel with command line filters to create a library of sharable analyst knowledge akin to what is embedded in Snort's well established set of signatures. Such a library would permit users to examine and quickly remove legitimate traffic and known slices of malicious activity and thereby allow them to focus on the more interesting remainder. The semantic knowledge required to filter traffic may also prove very useful to highlight activity of interest. In some instances, the user may wish to use this same library to encode their visualizations with the additional semantic information (e.g. highlight patterns matching the most recent virus outbreak) instead of immediately filtering. As the remaining data flows from the filtering stage it passes to the graphics engine and is subsequently displayed. Matching these data flows with efficient and effective visualizations has been partially explored, but with a near infinite range of potential visualizations and interaction techniques, there remain many areas of future work. A good starting point is to explore existing information visualization techniques from other domains and applying them to security problems. When such systems are built, they must be validated with more precise techniques than just anecdotal evidence and intuition. The quantitative evaluation presented in this dissertation is an

initial investigation into this area, but future systems should extend this work by formally exploring system effectiveness with candidate users. Likewise, visualization is a computationally intensive task. Hardware performance has only been explored at the most fundamental level. Further work is needed to determine the strengths and limitations of both the visualization hardware and the human using it. Finally, visualization provides a new means for creating both logs and reports that are more effective at communicating results to analysts, managers, end-users and customers. Little work has been done in this area and it has a great deal of potential for advancement.

REFERENCES

[1] Activeworx.org. "Honeynet Security Console." http://www.activeworx.org/programs/hsc/index.htm, last accessed 23 March 2006.

[2] Mustaque Ahamad, Wenke Lee, Ling Liu, Leo Mark, Edward Omicienski, Calton Pu and Andre Dos Santos. "Guarding the Next Internet Frontier: Countering Denial of Information Attacks." Proceedings of the New Security Paradigms Workshop; pp 136-143; September 2002.

[3] Air War College. "Army Battlefield Deception Operations." United States Air Force. http://www.au.af.mil/au/awc/awcgate /army/batdecep.htm, last accessed 15 February 2005.

[4] William Arbaugh, David Farber, and Jonathan Smith. "A Secure and Reliable Bootstrap Architecture."' Proceedings of the IEEE Symposium on Security and Privacy, pp. 65-71, May 1997.

[5] Stefan Axelsson. "Combining a Bayesian Classifier with Visualisation: Understanding the IDS." ACM Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec), 2004.

[6] Michael Bach. Fifty-two Optical Illusions and Visual Phenomena. http://www.michaelbach.de/ot/index.html, last accessed 23 March 2006.

[7] David Bailey. "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers." Supercomputing Review, August 1991, pp. 54-55.

[8] Edward Balas. "Honeynet Data Analysis: A technique for correlating sebek and network data." http://www.dfrws.org/bios/day2/Balas_Honeynets_for_DF.pdf, last accessed 5 January 2005.

[9] Dirk Balfanz, Glenn Durfee, Rebecca Grinter, D. K. Smetters and Paul Stewart. "Network-in-a-Box: How to Set Up a Secure Wireless Network in Under a Minute." USENIX Security Symposium, 2004.

[10] Robert Ball, Glenn Fink, Anand Rathi, Sumit Shah, and Chris North. "Home-Centric Visualization of Network Traffic for Security Administration." ACM Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec), 2004.

[11] Ben Bederson, James Hollan, Ken Perlin, Jonathan Meyer, David Bacon and George Furnas. "Pad++ A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics." Journal of Visual Languages and Computing, vol. 7, no. 1, 1996, pp 3-31.

[12] BeeSync.com. " PacketX - wrapper for WinPcap packet capture library.." http://www.beesync.com/packetx/index.html, last accessed 23 March 2006.

[13] Edwin Blake. "An Extended Platter Metaphor for Effective Reconfigurable Network Visualization." 8th International Conference on Information Visualization (IV), 2004.

[14] Yoram Bonneh, Alexander Cooperman and Dov Sagi. "Motion Induced Blindness." Nature, vol. 411, 2001, pp. 798–801.

[15] Darren Bounds. Packit - Network Injection and Capture. http://www.obtuse.net/software/packit/, last accessed 23 March 2006.

[16] Stuart Card, Thomas Moran and Allen Newell. The Psychology of Human Computer Interaction. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1983.

[17] Bill Cheswick and Hal Burch. The Internet Mapping Project. http://research.lumeta.com/ches/map/, last accessed 23 March 2006.

[18] Gregory Conti. Study evaluation documents are available at http://www.rumint.org/gregconti/publications/20060131_Evaluation_documents.zip, last accessed 23 March 2006.

[19] Gregory Conti. "Why Computer Scientists Should Attend Hacker Conferences." Communications of the ACM, March 2005.

[20] Robert Cook. "Visual Perception." From Comparative Psychology: A Handbook. Garland Publishing. Article available online at http://www.pigeon.psy.tufts.edu/ecp.htm, last accessed 23 March 2006.

[21] The Cooperative Association for Internet Data Analysis (CAIDA). http://www.caida.org/, last accessed 23 March 2006.

[22] The Cooperative Association for Internet Data Analysis. "Code Red Worm Infections." http://www.caida.org/tools/visualization/walrus/examples/codered/, last accessed 23 March 2006.

[23] Phil Craven. "Google's PageRank Explained and How to Get the Most Out of It." http://www.webworkshop.net/pagerank.html, last accessed 23 March 2006.

[24] Cybergeography.org. "An Atlas of Cyberspaces." http://www.cybergeography.org/atlas/atlas.html, last accessed 23 March 2006.

[25] Anita D'Amico, D. Tesone, Kirsten Whitely, B. O'Brien, M. Smith and E. Roth. "Understanding the Cyber Defender: A Cognitive Task Analysis of Information Assurance Analysts." Report CSA-CTA-1-1 , Air Force Research Laboratory, 2005. [FOUO]

[26] Anita D'Amico and Michael Kocka. "Information Assurance Visualizations for Specific Stages of Situational Awareness and Intended Uses: Lessons Learned." Visualization for Computer Security (VizSec), 2005.

[27] DataRescue.com. "IDA Pro Disassembler - multi-processor, windows hosted disassembler and debugger." http://www.datarescue.com/idabase/, last accessed 23 March 2006.

[28] Defense Information Systems Agency. "PKI Overview." http://www.disa.mil/apps/mgs/html/pki_overview.html, last accessed 22 June 2004.

[29] Disinfopedia.org. "Propaganda.." http://www.disinfopedia.org/wiki.phtml?title=Propaganda, last accessed 23 March 2006.

[30] Benjamin Edelman. "Empirical Analysis of Google SafeSearch." Berkman Center for Internet & Society, Harvard Law School, 14 April 2003.

[31] Benjamin Edelman. "Web Sites Sharing IP Addresses: Prevalence and Significance." Berkman Center for Internet & Society, Harvard Law School, 12 September 2003.

[32] Robert Erbacher, Kenneth Walker and Deborah Frincke. “Intrusion and Misuse Detection in Large-Scale Systems.” Computer Graphics and Applications, Vol. 22, No. 1, January/February 2002, pp. 38-48.

[33] Robert Erbacher and Deborah Frincke. “Visual Behavior Characterization for Intrusion and Misuse Detection.” Conference on Visual Data Exploration and Analysis VIII (SPIE), 2001.

[34] EtherApe: A Graphical Network Monitor. http://etherape.sourceforge.net/, last accessed 23 March 2006.

[35] Ethereal: A Network Protocol Analyzer. http://www.ethereal.com/, last accessed 23 March 2006.

[36] Glenn Fink, Robert Ball, Nipun Jawalkar, Chris North and Ricardo Correa. “Network Eye: End-to-End Computer Security Visualization.” ACM CCS Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec) 2004.

[37] Glenn Fink, Paul Muessig and Chris North. “Visual Correlation of Host Processes and Network Traffic.” Visualization for Computer Security (VizSec), 2005.

[38] Fyodor. Top 75 Network Security Tools List. http://www.insecure.org/tools.html, last accessed 23 March 2006.

[39] Simson Garfinkel and Robert Miller. “Johnny 2: a user test of key continuity management with S/MIME and Outlook Express.” Symposium on Usable Privacy and Security (SOUPS), 2005.

[40] Luc Girardin. “An Eye on Network Intruder-Administrator Shootouts.” USENIX 1st Workshop on Intrusion Detection and Network Monitoring, 1999.

[41] Al Globus and Eric Raible. “Fourteen Ways to Say Nothing with Scientific Visualization.” Computer, Vol. 27, No. 7, pp. 86-88, 1994.

[42] Tom Goldring. “Scatter (and other) Plots for Visualizing User Profiling Data and Network Traffic.” ACM Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec), 2004.

[43] John Goodall. The Intrusion Detection Tool Kit (IDtk). http://userpages.umbc.edu/~jgood/, last accessed 23 March 2006.

[44] John Goodall, Wayne Lutters and Anita Komlodi. "The Work of Intrusion Detection: Rethinking the Role of Security Analysts." Tenth Americas Conference on Information Systems, 2004.

[45] Google.com. "Google Technology." http://www.google.com/technology/, last accessed 23 March 2006.

[46] Google.com. "Googlebot: Google's Web Crawler.." http://www.google.com/webmasters/bot.html, last accessed 23 March 2006.

[47] Grace Project Homepage. http://plasma-gate.weizmann.ac.il/Grace/, last accessed 23 March 2006.

[48] Paul Graham. "A Plan for Spam." August, 2002. http://www.paulgraham.com/spam.html, last accessed 23 March 2006.

[49] Paul Graham. "Better Bayesian Filtering" January, 2003. http://www.paulgraham.com/better.html, last accessed 23 March 2006.

[50] Paul Graham. "So Far, So Good." August, 2003. http://www.paulgraham.com/sofar.html, last accessed 23 March 2006.

[51] GraphViz.org. "Graphviz - Graph Visualization Software." http://www.graphviz.org/, last accessed 23 March 2006.

[52] Katie Hafner. "Real Queasiness in Virtual Reality." The New York Times Online, November, 19, 1998.

[53] Grant Hammond,. The Mind of War, Smithsonian Institution Press, July 2004.

[54] G. Hardin. "Photosensitive Epilepsy." Epilepsy Matters, vol. 9, no. 3, Summer 1998.

[55] Christopher Healey. Perception in Visualization. Department of Computer Science, North Carolina State University. http://www.csc.ncsu.edu/faculty/healey/PP/, last accessed 23 March 2006.

[56] The Honeynet Project. "Scan of the Month." http://www.honeynet.org/scans/, last accessed 23 March 2006.

[57] Michael Howard and David LeBlanc. Writing Secure Code. Microsoft Press, 2002.

[58] Jim Hu. "House Approves Anti-Spam Legislation." CNET News, 18 July 2000. http://news.com.com/2100-1023-243310.html, last accessed 23 March 2006.

[59] Darrell Huff. How to Lie With Statistics. W. W. Norton and Company, 1993.

[60] IEEE Visualization Conference. http://vis.computer.org, last accessed 23 March 2006.

[61] Alftred Inselberg. "Multidimensional Detective." IEEE Symposium on Information Visualization, 1997.

[62] Tom Jagatic, Nathaniel Johnson, Markus Jakobsson, and Filippo Menczer. "Social Phishing." Indiana University: Bloomington - Working Paper. http://www.informatics.indiana.edu/fil/papers.asp, last accessed 23 March 2006.

[63] Gerald Jones. How to Lie With Charts. Authors Choice Press, 2000.

[64] Jukka Juslin. "Intrusion Detection and Visualization Using Perl." O'Reilly Open Source Conference, 2001.

[65] Hideki Koike and Kazuhiro Ohno. "SnortView: Visualization System of Snort Logs." ACM Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec), 2004.

[66] Anita Komlodi, John Goodall and Wayne Lutters. "An Information Visualization Framework for Intrusion Detection." ACM Conference on Human Factors in Computing Systems (CHI), 2004.

[67] Anita Komlodi, Penny Rheingans, Utkarsha Ayachit, John Goodall and Amit Joshi. "A User-centered Look at Glyph Based Visualization." Visualization for Computer Security (VizSec), 2005.

[68] Eleftherios Koutsofios, Stephen North, Russell Truscott, Daniel Keim. "Visualizing large-scale telecommunication networks and services (case study)." IEEE Visualization, 1999.

[69] Stephen Lau. "The Spinning Cube of Potential Doom." Communications of the ACM, Vol. 7, No. 46, June 2004.

[70] Elias Levy. "Interface Illusions." IEEE Security and Privacy, vol. 2, no. 6, November/December 2004, pp. 66-69.

[71] Lyman, Peter and Varian, Hal. "How Much Information 2003." http://www.sims.berkeley.edu/how-much-info-2003, last accessed 23 March 2006.

[72] Lyman, Peter and Varian, Hal. "How Much Information 2000." http://www.sims.berkeley.edu/how-much-info, last accessed 23 March 2006.

[73] MailAbuse.org. "The Mail-Abuse Project." http://www.mail-abuse.org/, last accessed 23 March 2006.

[74] David Marchette. Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint, Springer, 2001.

[75] Jonathan McPherson, Kwan-Liu Ma, Paul Krystosek, Tony Bartoletti, Marvin Christensen. "PortVis: A Tool for Port-Based Detection of Security Events." ACM Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec), 2004.

[76] George Miller. "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information." The Psychological Review, vol. 63, 1956, pp. 81-97.

[77] Charles Morris and Albert Maisto. Psychology: An Introduction. 10th Edition, Prentice Hall. Summary available online at http://cwx.prenhall.com/bookbind/pubbooks/morris2/chapter3/medialib/summary/1.html, last accessed 23 March 2006.

[78] Tamara Munzner. "Interactive Visualization of Large Graphs and Networks." Ph.D. dissertation, Stanford University, June 2000.

[79] Jakob Nielsen and Thomas Landauer. "A Mathematical Model of the Finding of Usability Problems." ACM Conference on Human Factors in Computing Systems, 1993.

[80] Kofi Nyarko, Tanya Capers, Craig Scott and Kemi Ladeji-Osias. "Network Intrusion Visualization with NIVA, an Intrusion Detection Visual Analyzer with Haptic Integration." 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002.

[81] ORDB.org. "The Open Relay Database." http://www.ordb.org/, last accessed 23 March 2006.

[82] OSSIM.net. "Open Source Security Information Management.." http://www.ossim.net/, last accessed 23 March 2006.

[83] Henry Owen. "Georgia Tech Honeynet Research Project." http://users.ece.gatech.edu/~owen/Research/HoneyNet/HoneyNet_home.htm, last accessed 23 March 2006.

[84] Larry Page, Sergey Brin, Rajeev Motwani and Terry Winograd. "The PageRank Citation Ranking: Bringing Order to the Web." Stanford Digital Library Technologies Project, 1998.

[85] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson and Larry Peterson. "Characteristics of Internet Background Radiation." ACM SIGCOMM Internet Measurement Conference (ACM-IMC), 2003.

[86] Partick Pantel and Dekang Lin. "SpamCop-- A Spam Classification & Organization Program." AAAI Workshop on Learning for Text Categorization, 1998.

[87] Samuel Patton, William Yurcik, and David Doss. "An Achilles' Heel in Signature-Based IDS: Squealing False Positives in SNORT." Recent Advances in Intrusion Detection (RAID), 2001.

[88] Thomas Ptacek and Timothy Newsham. “Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection.” Secure Networks Inc., 1998.

[89] Bernice Rogowitz, LLoyd Treinish and Steve Bryson. “How Not to Lie With Visualization.” Computers in Physics, Vol. 10, No. 3, May/June 1996, pp. 268-273.

[90] RUMINT.org. “RUMINT: A PVR for Network Traffic and Security Visualization..” http://www.rumint.org/, last accessed 23 March 2006.

[91] RUMINT.org. .http://www.rumint.org/software/rumint/rumint_v.192.zip, last accessed 23 March 2006.

[92] Mehram Sahami, Susan Dumais, David Heckerman and Eric Horvitz. “A Bayesian Approach to Filtering Junk E-Mail.”' AAAI Workshop on Learning for Text Categorization, 1998.

[93] “Security Incident Fusion Tool.” National Center for Advanced Secure Systems Research Group. http://www.ncassr.org/projects/sift/papers/, last accessed 23 March 2006.

[94] The Shmoo Group. “Capture the RootFu!” http://www.shmoo.com/cctf/, last accessed 19 November 2004.

[95] Sniph. “Snot.” Security Focus, http://www.securityfocus.com/tools/1983, last accessed 23 March 2006.

[96] Snort.org. “Snort: The Open Source Intrusion Detection System.” http://www.snort.org/, last accessed 23 March 2006.

[97] The SpamAssassin Project. http://spamassassin.apache.org/, last accessed 23 March 2006.

[98] Spam Conference ’03 Proceedings. Cambridge, MA, 2003. http://spamconference.org/proceedings2003.html, last accessed 23 March 2006.

[99] SpamProbe: A Fast Bayesian Spam Filter. http://spamprobe.sourceforge.net/, last accessed 23 March 2006.

[100] Robert Spence. Information Visualization. ACM Press, 2001.

[101] Frank Swiderski and Window Snyder. Threat Modeling. Microsoft Press, 2004.

[102] SubmitExpress.com. "Submit Express - Search Engine Placement. " http://www.submitexpress.com/, last accessed 23 March 2006.

[103] Submit-It.com. "Search Engine Optimization Tips." http://www.submit-it.com/subopt.htm, last accessed 23 March 2006.

[104] Soon Tee Teoh, Kwan Liu Ma, S. Felix Wu. "Case Study: Interactive Visualization for Internet Security." IEEE Information Visualization, 2002.

[105] Soon Tee Teoh. "Graphical Presentation of Stepping-Stone Pairs Found." Initial Results. http://graphics.cs.ucdavis.edu/ ~steoh/ research/tcpdump/tcpdump.html, last accessed 12 April 2004.

[106] Soon Tee Teoh, Kwan Liu Ma and S. Felix Wu. "A Visual Exploration Process for the Analysis of Internet Routing Data." IEEE Information Visualization, 2003.

[107] Soon Tee Teoh, T. J. Jankun-Kelly, Kwan-Liu Ma, and S. Felix Wu. "Visual Data Analysis for Detecting Flaws and Intruders in Computer Network Systems." IEEE Computer Graphics and Applications, IEEE Computer Society Press, September/October, 2004.

[108] TCPDump.org "TCPDump Public Repository." http://www.tcpdump.org/, last accessed 23 March 2006.

[109] Graziella Tonfoni. "On Developing teaching material for an Information Design Course." Technical Report produced for the College of Library and Information Services at University of Maryland, College Park, 1997.

[110] Edward Tufte. The Cognitive Style of PowerPoint. Graphics Press, 2003.

[111] Edward Tufte. Envisioning Information. Graphics Press, 1990.

[112] Edward Tufte. Power Point is Evil. Wired Magazine Online, http://www.wired.com/wired/archive/11.09/ppt2_pr.html, last accessed 23 March 2006.

[113] Edward Tufte. The Visual Display of Quantitative Information. Second Edition. Graphics Press, 2001.

[114] Edward Tufte. Visual Explanations: Images and Quantities, Evidence and Narrative. Graphics Press, 1997.

[115] United States Federal Trade Commission Spam Forum. Washington, DC, 2003. http://www.ftc.gov/bcp/workshops/spam/, last accessed 23 March 2006.

[116] United States Military Academy. "CyberDefense Exercise." http://www.itoc.usma.edu/cdx/, last accessed 23 March 2006.

[117] Vipul's Razor. http://razor.sourceforge.net/, last accessed 23 March 2006.

[118] Fred Von Lohman. "Meditations on Trusted Computing;" Electronic Frontier Federation; http://www.eff.org/Infra/trusted_computing/20031001_meditations.php, last accessed 23 March 2006.

[119] Yair Wand and Richard Wang. "Anchoring Data Quality Dimensions in Ontological Foundations." Communications of the Association for Computing Machinery, November 1996.

[120] Ke Wang and Salvatore Stolfo. "Anomalous Payload-based Network Intrusion Detection." Seventh International Symposium on Recent Advances in Intrusion Detection, 2004.

[121] Alma Whitten and J. Doug Tygar. "Why Johnny can't encrypt: A usability evaluation of PGP 5.0." USENIX Security Symposium, 1999.

[122] Christopher Williamson and Ben Shneiderman. "The dynamic HomeFinder: evaluating dynamic queries in a real-estate information exploration system." ACM Conference on Research and Development in Information Retrieval (SIGIR), 1992.

[123] Leland Wilkinson. The Grammar of Graphics. Springer-Verlag, 1999.

[124] Windows Packet Capture Library. http://winpcap.polito.it/, last accessed 23 March 2006.

[125] Wired.com. "Man Arrested for Spam Rage." Wired News, 21 November 2003. http://www.wired.com/news/culture/0,1284,61339,00.html, last accessed 23 March 2006.

[126] Yahoo: How to Suggest a Site. http://docs.yahoo.com/info/suggest/, last accessed 23 March 2006.

[127] Xiaoxin Yin, William Yurcik, Michael Treaster, Yifan Li and Kiran Lakkaraju. "VisFlowConnect: NetFlow Visualizations of Link Relationships for Security Situational Awareness." ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC), 2004.

[128] InSeon Yoo. "Visualizing Windows Executable Viruses Using Self-Organizing Maps." ACM Workshop on Visualization and Data Mining for Computer Security (VizSec/DMSec), 2004.

[129] William Yurcik, James Barlow, Kiran Lakkaraju, and Mike Haberman. "Two Visual Computer Network Security Monitoring Tools Incorporating Operator Interface Requirements." ACM CHI Workshop on Human-Computer Interaction and Security Systems (HCISEC), 2003.

[130] Michal Zalewski. "Strange Attractors and TCP/IP Sequence Number Analysis." http://www.bindview.com/Services/Razor/Papers/2001/tcpseq.cfm, last accessed 23 March 2006.

[131] Michal Zalewski. "Strange Attractors and TCP/IP Sequence Number Analysis - One Year Later." http://lcamtuf.coredump.cx/newtcp/, last accessed 23 March 2006.

The views expressed in this dissertation are those of the author and do not reflect the official policy or position of the United States Military Academy, the Department of the Army, the Department of Defense or the United States Government.